EVALUATING THE EFFECTS OF MODEL SIMPLIFICATIONS ON THE TRANSFERENCE OF POLICIES LEARNED IN SIMULATION

A Dissertation
Presented to
The Academic Faculty

By

Patrick S. Meyer

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology

December 2020

EVALUATING THE EFFECTS OF MODEL SIMPLIFICATIONS ON THE TRANSFERENCE OF POLICIES LEARNED IN SIMULATION

Approved by:

Prof. Dimitri Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Eric Feron
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Daniel Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Michael Steffens
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Gian Luca Mariottini
Autonomous Systems
Draper Laboratory

Date Approved: August 31, 2020

In theory, there is no difference between theory and practice. In practice, there is.

Benjamin Brewster

ACKNOWLEDGEMENTS

They say completing a PhD is among the loneliest of endeavors and is all about personal accomplishment. While I certainly identify with the first part, the second part could not be further from the truth. This document, and all of the ideas I have tried to express within it would not be possible without a little (read: a lot) of help from my friends.

First, I would like to thank my advisor, Professor Mavris, for giving me the opportunity to better myself through this process. Your support and trust in my abilities was invaluable. Throughout all of this work, you've put in countless hours of discussions and pushed me with your questions. Thank you for everything you have done and all of the care you provide your students.

I would also like to thank all of the members of my committee. Through all of our discussions, you've pushed me to explore and go beyond where I thought my limits were. Your feedback kept me from wandering too far down countless rabbit holes. Your understanding and willingness to share guidance is greatly appreciated. To Professor Schrage, Professor Feron, Dr. Steffens, and Dr. Mariottini, I want to sincerely thank you for all the time and effort you've given to me.

To Tanya, Adrienne, and the rest of the ASDL staff, thank you for all of the support you provide the students. I can't imagine the difficulties you face in wrangling all of us through our work. Please know that all of your effort and understanding is greatly appreciated.

To Coline, Ethan, Sam, and the rest of the ADEPT and Marine Robotics crew, I can't overstate how much you have helped me through all of this. Thank you for giving me the space to bounce ideas off of, talking through every last nitty-gritty detail, and telling me when you thought I was being an idiot. You deserve much of the credit and none of the blame for this work. The time we've spent playing with robots, competing around the world, and learning together has kept me sane through all of this.

To Daniel, Kelly, and Michael, thank you for convincing me that pursuing a PhD was

not the worst decision in the world. Your guidance and support throughout this process made all of this possible. The atmosphere you promoted, random philosophical discussions, and the constant new issues to overcome with our robots helped keep me grounded through this whole journey. Your guidance truly inspired me to pursue all of this.

To my parents, I would not be here without you and all of the support you've given me. I'm incredibly lucky to have had you in my life. You've made sure I had the right balance of guidance and space needed to grow as a person. Your immeasurable love and support have gotten me here, and I can't thank you enough.

And finally to Danica, thank you for being so understanding and so patient with me. I'm sure dating a doctoral student can't be the most enjoyable experience. You've dealt with me turning little things into deep philosophical questions of meaning, my complete lack of consistent sleep schedule, and an utter lack of separation between work and home lives. Thank you for your unending patience, support, and love. You kept me at it even when the challenges seemed insurmountable. Thank you for seeing this through with me, and I can't wait to see where we go next.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

As technology advances, we are constantly being pushed to search new frontiers. One of the most fundamental changes occurring today is the continued push for unmanned and autonomous systems. These systems are becoming more prevalent in everyday life. Both the military and civil worlds are being transformed by the development and deployment of unmanned systems to a wider range of scenarios. From the battlefield to the manufacturing line, the slow march of progress is obvious. Even in more subtle ways, like the pricing of goods and identification of strategic information, systems without a human as the primary drive of decision making are advancing.

As this field has grown and matured, it has continuously advanced towards increasing levels of autonomy. That is, systems that once could execute only rigid routines are able to handle more and more abstract tasks. As an example, the cars of today have gone from *maintain this speed* to *drive on this highway*. As this push towards greater and greater levels of autonomy continues, new methods for developing policies of control are required. As goals become more abstract, the number of edge cases that must be considered explodes. Classically derived and rule based policies quickly become too complex for practical implementations.

Recent breakthroughs in reinforcement learning hope to address this problem. The primary advantage of reinforcement learning based systems is their focus on goal driven behavior. That is, first the behavior of a system is defined in a goal oriented framework. Positive outcomes, such as winning a game, delivering a package, or identifying a target, are rewarded. Negative outcomes, such as crashing a car, missing a target, or failing to complete a task are penalized. Policies can then be discovered and refined that accumulate positive experiences and avoid negative experiences through exploration.

In developing reinforcement learning based policies, modeling and simulation has been an indispensable tool. Systems can be subjected to scenarios that would be too costly,

dangerous, or simply impossible to recreate and their behavior rigorously evaluated. This allows for the exploration necessary to identify and improve policies that will work in a given environment. However, modeling a system necessarily introduces epistemic uncertainties in the predictions produced due to simplifications made in the modeling process. These uncertainties can produce complex effects resulting in behaviors that aren't seen in the real system or the suppression of subtle behaviors that expand to become more significant. This mismatch between simulated and real experience is often attributed to the so-called *reality gap*.

An entire sub-field of autonomous systems research, called sim-to-real, has attempted to address this gap. Common approaches to producing policies that transfer from the simulated environment to the target environment include use of high-fidelity simulation and techniques from transfer learning. While some of these results have been promising, there remain gaps in our understanding of the role of modeling choices on transference between environments. Namely, the current literature does not directly address the issue of selecting phenomena to represent in the simplified model of a system.

To address this, a method for comparing the relative importance of phenomena to be considered for a model of a system is developed. By representing systems as a collection of phenomena, the possible simplification space can be represented as a discrete set of models to consider. A sampling based approach is implemented to explore this space. This is used to sample a representative portion of the possible simplifications that can be made in representing a system. Experiments show that a relatively small sampling of the space allows for phenomena to be ordered in a way such that significantly simplified versions of a full referent model maintain similar transference properties.

By evaluating individual sampled simplifications, the importance of the distinct phenomena can be quantified. This is done by grouping simplifications into non-exclusive sets. These sets are characterized by individual phenomena, and allow for an aggregated measure of the sets' performance to represent the importance of each characteristic phe-

nomena. It is shown that using this measure of importance leads to the development of an ordered set of simplified models with increasing fidelity. When compared to alternative methods of generating similar sets of models, the set identified through this method shows improved transference at lower fidelity levels.

Additional experiments show that this method is applicable in cases where perfect knowledge of the referent system doesn't exist. In such a way, this shows that it can be applicable to realistic systems where evaluations of transference to the true system are impossible. Choices in sampling strategy and metrics for evaluation of individual phenomena are also evaluated. A final experiment showing applications to a common reinforcement learning benchmark, the Acrobot system, show that the method is largely successful, correctly identifying one phenomena that qualitatively changes the required control for success as the most important. Similarly, it identifies a distracting phenomena that has negligible affect on system behavior under the trained policy as the least important. Even more encouragingly, some of the simplified models identified through the proposed method actually produce policies that outperform those trained directly on the true system.

CHAPTER 1
INTRODUCTION

Unmanned systems have become more prevalent throughout everyday life. Both the military and civil worlds have shown increasing interest in developing and deploying unmanned systems to a wider range of scenarios. These systems have matured to the point of potentially replacing manned systems in the foreseeable future in some fields. From manufacturing to warfighting, the increase in unmanned systems has fundamentally changed how many things are done.

As the field of unmanned systems has grown and matured, it has continuously advanced towards increasing levels of autonomy. [24] Autonomous cars have entered the mainstream public knowledge, and are now often seen as an inevitable next step towards the future of transportation. Advances are allowing manufacturing to not only use automatic routines with robots, but more complex scenarios such as part picking in cluttered environments. Algorithms decide the prices of goods purchased throughout many marketplaces. Unmanned vehicles have allowed the military to remove people from dirty and dangerous situations, with autonomous systems beginning to fill the gap of dull missions as well. [13, 24, 50, 72]

While unmanned systems in general have seen expanded use, the next frontier for these systems will be the development of greater autonomous capabilities. The US Department of Defense has called autonomous technologies one of the necessary enablers for future systems. [24] Car manufacturers are in a race to develop autonomous capabilities for the next generation of transportation infrastructure. Warehouses and manufacturing plants are also looking into increasing the autonomous capabilities of robotic equipment. This thesis will explore the challenges of developing behavioral policies for these autonomous systems to employ. Namely, a common challenge of developing these behavioral policies is transferring them from simulated environments to the real world. This thesis will develop an

approach to identifying critical components of these simulations for this transference.

1.1 Modern Autonomous Systems

While the promises of unmanned and autonomous systems are quite attractive, there are dangers as well. Their core strength, reducing the need for human intervention, can also be a major weakness. While often taken for granted, basic levels of common sense allow for humans to avoid possibly disastrous results even in areas where they have less than expert level experiences. Consider driving. Even relatively new drivers will quickly apply the brakes if an obstacle appears unexpectedly or take preventative measures if they are presented with a situation they have never encountered before. This is because a human operator can pull from experiences that are not directly relevant to the current task, and generalize their actions. Currently available autonomous systems have shown little ability for this generalizable behavior.

Various definitions of autonomy have been proposed to frame these desired generalizable behavior. One of the most popular is the current SAE standard for autonomous vehicles. [116] This proposes a taxonomy based on 6 levels of autonomy, ranging from Level 0, fully manual, to Level 5, fully autonomous. The main differentiating feature in this case is the level of abstraction in describing the goals and capabilities of the system. Control of low-level functionality, such as maintaining a speed, etc., would be considered at a low level of autonomy, often called an "automatic" system. As a system progresses through the levels, it can handle more abstract goals, such as *go to this destination* or *pick up this person*. These levels of autonomy are a useful basis for a wide range of systems. Many current systems would be classified as "automatic," meaning they have predefined routines that can be executed with little leeway for variance in their environments. The most advanced of these systems may be able to detect when an unexpected event or scenario is encountered and shutdown. An example of this would be many driving assist modes that return control to the human operator when an issue is detected. These systems can have

catastrophic consequences though, especially when the human operator is unclear on the limits of autonomy. [50]

Other groups are also considering the difficulty in developing autonomous systems. ASTM International has produced work specifically to address issues that arise form the application of autonomy for aerospace systems. [18] As was noted, there are additional challenges that must be faced by autonomous aircraft, as the common "stop and reset" strategy is no longer viable. As such, possible hierarchical architectures for control that allow for alternative safety measures have been suggested. [109] This standard allows for complex functions, such as modern neural network architectures, to be used in the decision making process by wrapping them in a safety manager that can switch to a rigorously derived controller to avoid and recover from unsafe scenarios.

While this and other high level architectures are necessary for developing safe systems, they assume the existence of a complex control function to be wrapped in the first place. The process of developing these complex control functions is still an emerging field and will be the primary focus of this work. Some modern approaches are beginning to show progress, with current techniques often applicable to multiple problems. [71, 78] However, these techniques often must be implemented multiple times to develop distinct behaviors for each potential scenario. There remains significant work left in developing behaviors that are themselves generalizable.

One promising field for generating generalizable behavior is reinforcement learning. Reinforcement learning is a flexible approach to problem definition, where agents learn by exploring their environments and receiving rewards or punishments depending on their actions. [112] This framework is loosely inspired by biological systems, and allows for very general problem statements. Many recent examples in the reinforcement learning community have shown ground breaking results, from superhuman performance at arcade games [78] to beating humans at complex board games often thought impossible for a computer to solve, like Go [107]. These breakthroughs have shown reinforcement learning

to be an attractive area for further research. As such, this thesis will focus on reinforcement learning for its ability to train behavioral policies.

1.2 Reinforcement Learning in the Real World

Reinforcement learning has led to many recent breakthroughs. Policies for playing arcade games at superhuman levels have been developed [78], board games once thought impossible for a computer to solve have been conquered [107], even collaborative games such as capture the flag have seen the successful application of reinforcement learning [52]. However, a common thread of many of these breakthroughs is application in a virtual domain. Video and board games can both perfectly be represented in a virtual environment.

While virtual environments can present difficult challenges, systems that must operate in the real world present unique difficulties. [64] This is because one of the main requirements for finding a reasonable policy through reinforcement learning is having sufficient exploration of the state and action spaces the system can operate in. [119] The basic theory behind this approach is to increase the scenarios the autonomous agent has been exposed to, thereby increasing the likelihood that even rare scenarios have been considered during training. Similar techniques for gathering large datasets have seen extended use in other areas of machine learning problems, such as image classification, with great success. [26, 38]

For embodied systems interacting with the real world, this presents a very real danger. The rare scenarios that must be considered may pose a threat to the platform or to bystanders. As a concrete example, consider training an autonomous agent for car driving. To develop a relatively robust controller, the car would have to be put into dangerous situations, such as operating near the limits of physical control, if it is to be able to recover from them. This represents an unacceptable risk, as any training would now become prohibitively expense and dangerous. Similarly, even having access to these rare and impactful events may be difficult to guarantee. They are by definition rare, after all.

While this situation sounds quite dire, many reinforcement learning based policies have been used on systems in the real world today. [15, 88, 120] How is this possible? Most training of autonomous agents today leverage two approaches. First and simplest is through human examples. [35] Human operators control the targeted systems under normal operating conditions, generating significant data for an autonomous agent to consider. While this approach can provide a significant source of data, it does not address the above issues where rare and dangerous situations need to be reconstructed. The second approach addresses this through the use of modeling and simulation. At this point, nearly any agent will first be trained in a simulated environment. While these may be computationally expensive to run, they are significantly cheaper and infinitely safer than allowing a naive agent to interact with the real world.

While the use of simulation to provide data for rare or dangerous situations is almost necessary, it presents its own downsides. Simulated environments must necessarily make simplifications that alter the behavior of both the agent being trained, and the environment's responses to that agent's actions. [96] Inaccuracies in both the environment and agent models compound, leading to what is often called the *reality gap*. This gap presents many problems leading to policies failing to transfer from the simulated environment to the real world. Some approaches to this problem have been suggested in the literature, but it remains one of the most difficult problems when it comes to developing behavioral policies through reinforcement learning. [15, 50, 64, 120]

1.3 Research Objective

While reinforcement learning represents an attractive approach to developing the next stage of behavior for autonomous systems, it is still an emerging field of research. There are significant open gaps that must be addressed. Particularly, most frameworks for conducting reinforcement learning require significant exploration of their possible state and action spaces. This is fine for many of the virtual domains that reinforcement learning has been

applied to successfully, but will certainly cause issues for systems that operate in the real world. Training an autonomous car from scratch in the real world would be prohibitively expensive, not to mention dangerous.

As such, a significant amount of work has gone into developing reinforcement learning behaviors for real systems in simulations. This allows for many benefits, including enhanced safety, ease of recreation of failure cases, and possibly increasing speed of training through parallelization. However, systems trained in simulated environments often fail to transfer to the real world. There are varied reasons for this that will be explored throughout this dissertation, but this provides the main motivation for this thesis. As such, the main motivating objective for this work can be stated as:

Motivating Objective: *Identify possible methods for the improvement of models used in simulation-based training of reinforcement learning derived policies.*

1.4 Document Structure

This dissertation is organized into 6 chapters. This chapter briefly discussed autonomy as a major developing technology for a wide range of industries. Reinforcement learning was identified as a emerging field of study that has seen significant breakthroughs in the recent literature. Reinforcement learning has been proposed as a general purpose framework for developing policies for autonomous behaviors. This chapter also discussed general trends in developing these policies, including simulation based training. This was used to frame the motivating objective for this work, identifying possible methods for the improvement of models used in simulation-based training of reinforcement learning derived policies.

Chapter 2 will discuss relevant background information to further develop this objective. Literature on modern reinforcement learning frameworks will be discussed to identify important trends and limitations. Work from the sim-to-real field will also be discussed to evaluate current methods for producing autonomous policies that can transfer from the

simulated world to the real world. Literature from the broader modeling and simulation community will also be reviewed. This review of the relevant literature will be used to identify gaps in the current state of the art and frame the development of a method to identify the relative importance of phenomena within simulation models.

Chapter 3 will further develop this method. The goal of the method is to identify a useful measure of importance, called *phenomena criticality*, that can be used to identify simplified models of a system to be used in simulations for developing transferable policies. Research questions and associated experiments will be proposed to evaluate the method. Chapter 4 will then develop a simple model development strategy employing these phenomena criticality measures. As before, the goal will be to construct models of a system for use in simulations to develop reinforcement learning based policies that are transferable to the truth system. Additional research questions and experiments will be proposed to evaluate the practical considerations of this method.

As the work presented in Chapter 3 and Chapter 4 will be tightly integrated, the results of their proposed experiments will be collected and presented in Chapter 5. This will cover how the method for measuring phenomena criticality and the resulting simplified models compare with baseline methods, possible alterations to the method, and impacts of imperfect information during evaluation.

Finally, Chapter 6 will collect the major points of each of the proceeding chapters. The overall research framework, methodology, and results will be summarized. Concluding thoughts including identification of specific contributions, limitations, and possible future research directions will be provided.

CHAPTER 2
BACKGROUND AND RELATED WORK

When looking at developing modern autonomous systems in the previous chapter, on of the most promising frontiers for research is in reinforcement learning. Unlike many previous paradigms for developing behavioral controls for autonomous systems, like analytically derived controllers or rule based systems, reinforcement learning based behaviors tend to be more flexible and their simple problem formulation allows them to be applied to many systems. [64] This flexibility comes both in their implementation and in the actual behaviors they result in, finding novel solutions that may not have been obvious before their use.

While promising, reinforcement learning based approaches still face significant challenges. As discussed briefly in the previous chapter, one of the greatest of these challenges is in training these policies. They require significant exploration of the possible state and action spaces for their target systems. As such, training directly on the desired system may be impossible, unsafe, or prohibitively expensive. To account for this, simulation techniques are often used in place of training on the true system. [64] While useful, these policies trained in simulation often fail to then transfer to the real world. [53, 64, 120] This led to the development of the motivating objective behind this thesis:

Motivating Objective: *Identify possible methods for the improvement of models used in simulation-based training of reinforcement learning derived policies.*

This chapter will review the relevant background work and identify gaps with respect to this objective in the current literature. This will focus on modern reinforcement learning methods, sim-to-real approaches, and modeling methods from other fields.

2.1 Modern Reinforcement Learning

While autonomous systems have seen significant development over the past decades, reinforcement learning specifically has started taking root as as a promising general purpose method for policy development. [64] This section will discuss a selection of recent contributions to the field of reinforcement learning for policy synthesis. While many of the modern frameworks for modern reinforcement learning have shown incredible results, they are often faced with a tradeoff between exploration of the state action space with exploitation of the already explored space. As such, this tradeoff and common approaches to addressing it will be discussed as well. Finally, this section will discuss some observations on common issues in reinforcement learning and specifically discuss the need for simulation based training for many systems.

2.1.1 Reinforcement Learning Frameworks

One of the major contribution of reinforcement learning is the flexible problem definition that allows for goals to be stated more naturally and with fewer assumptions. [112] The simplest way to imagine this problem is to consider the interaction between some agent and its environment. The agent takes various actions, which alter its state in the environment. The environment then responds to these actions and altering states by providing a reward signal to the agent. The goal of a reinforcement learning algorithm is to identify a *policy* to determine actions to take based off of the state of the agent that maximizes some running sum of the reward signal received.

As an example, consider training an agent to play the game of chess. On each turn, the agent's policy decides on a move. In this case, the policy would take in the current game state, the positions of all pieces on the board, and return the piece and target location of the move. The environment in this case is the game state. Once the agent's move has been played, the opponent responds with a move of their own, and the environment is updated to

the new game state. This new state is passed to the policy, and it continues to iterate from there.

In this example, their are many choices for possible reward signals. This choice in reward signal can have a significant impact on the resulting policies that are found. [81] One option is to provide a reward only at the end of the game. This reward would be either positive if the agent wins the game, or negative if the agent loses the game. This would be an example of a sparse reward signal, which are often harder to train on but may lead to more general solutions. [64] Other options, such as providing a reward based on capturing pieces, etc. would also be reasonable choices.

To formalize this, nearly all modern reinforcement learning problems are posed as a Markov Decision Process, or MDP. In this, we consider some environment, $\mathcal{E}$. $\mathcal{E}$ is defined by four major components: the possible state space, $\mathcal{S}$; the possible action space, $\mathcal{A}$; the state transition function, $\zeta : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$; and the reward signal, r. The state and action spaces can either be discrete or continuous spaces, and the action space may be limited depending on the current state. In general, the state transition function may be considered stochastic. As discussed above, the reward signal is an arbitrary function that maps from the state-action space to a scalar real number, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$. That is, the reward is dependent on the current state of the agent, the action taken by the agent, and the resulting next state of the agent.

In general, these decision processes are considered as discrete in time. That is, we can say the state, action, and resulting reward at timestep i are defined as $s_i \in \mathcal{S},\ a_i \in \mathcal{A},\ r_i \in \mathbb{R}$, respectively. Given this, we can define a trajectory of states and actions through the environment, τ. We can also define a return function, $\mathcal{R}$, that acts to consider the long term rewards that are achieved. This most often takes the form of $\mathcal{R} = \sum_{i \in \rho} \gamma^i r_i$ where $\gamma \in [0, 1]$ is called the discount factor. This common form attempts to solve what is called the *temporal credit assignment problem*. [112] That is, for environments with relatively sparse reward signals, it can be difficult to determine which actions actually led

to the positive result as no single action is fully responsible. The action just before the reward was only possible due to the series of actions that led the agent to that state. By distributing this reward backwards in time, the actions that led to the eventual payoff can also be rewarded. The discount factor then becomes a parameter that can be tuned to favor either short-term or long-term rewards.

Given this, we can define a policy, π that maps from the current state to a possible action, or $\pi : \mathcal{S} \rightarrow \mathcal{A}$. This policy may be stochastic, such that the return value is actually a distribution of actions to be taken, and is often a parameterized function. If we denote the parameter vector as θ, we can denote a specific implementation of that policy as π_θ.

We can now define the goal of a reinforcement learning problem. Considering the general case where the environment has a stochastic transition, we can define the expected return of a policy as:

$$J(\theta) = \mathbb{E}_{r,s\sim\mathcal{E},a\sim\pi_\theta}[\mathcal{R}] \tag{2.1}$$

That is, the value of a parameterization is equal to the expected return in the environment, $\mathcal{E}$, while acting under the policy π_θ The goal of a reinforcement learning problem can then simply be state as maximizing this expected return:

$$\arg\max_\theta J(\theta) \tag{2.2}$$

While simple and compact in form, this is a very powerful expression. Behaviors as diverse as driving an autonomous vehicle, such as in [20], to playing classic arcade games, such as in [78], can be encapsulated by this.

While clearly powerful in representation, the definition of this is only a portion of finding a useful policy. There are many different frameworks for solving this optimization problem. One of the most common modern frameworks is Q-learning. [40, 78, 131] The basic premise behind Q-learning is training an approximator of the the so called quality-function,

hence the name. This quality function takes the form $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. While similar in form to the reward signal, this is meant to approximate the expected total return the agent will receive for selecting an action at a given state. This is often done in a process called value iteration. This takes advantage of the form shown above for the return, $\mathcal{R}$, and the Bellman principle. The Bellman principle notes the iterative form of the return function, where if we consider the return following timestep i as $\mathcal{R}_i$, then we can state $\mathcal{R}_i = r_i + \gamma \mathcal{R}_{i+1}$. Replacing $\mathcal{R}$ with our estimator, $Q(s, a)$, we have $Q(s_i, a_i) = r_i + \gamma Q(s_{i+1}, \pi_\theta(s_{i+1}))$. In most cases, the policy then simply becomes $\pi_\theta(s) = \arg\max_a Q(s, a|\theta)$, with the parameter vector used to parameterize the Q-estimator.

Early works in this area often focused on providing mathematical proofs of convergence for policies and estimators under this framework. [131] As such, their results were largely limited to toy problems. This was due to the limited representations of the parameterized functions they were using at the time. Given the recent development of techniques for using artificial neural networks, such as breakthroughs with in computer vision problems like Convolutional Neural Networks [67], there was a desire to use these as function approximators instead.

Mnih et al showed that deep neural networks can be used as these function approximators, given a few tricks. [78] First, instead of using the normal on-policy training that was often used for reinforcement learning, they used an off-policy method through what they called a *replay buffer*. That is, instead of training the network at each time step as actions were selected, they stored the trajectory of states, chosen actions, rewards, and next states provided by the agent and environment in a buffer. By sampling experiences from this buffer, they could essentially replay past experiences to ensure old positive behaviors weren't forgotten. This also allowed them to use the experiences and the Bellman equation formulation of the expected return to train their approximator network in a supervised fashion, a more mature area of machine learning research. Other tricks, such as the use of a target network, were largely implement to stabilize the training of the network.

While this approach represented a huge step forward in terms of capabilities for reinforcement learning based policies, it had significant shortcomings. First, it is largely limited to discrete action spaces, such as those seen in the arcade games used in [78]. This is because the policy implementation, an $\arg\max$ problem, becomes intractable for continuous action spaces. While it's possible to discretize continuous action spaces to match this form, this will always have inherent limitations in representation. Additionaly, this may not solve the intractability problem for high dimensional action spaces. As such, Q-Learning based methods are often limited to virtual problems, or highly simplified problems.

One approach to handle continuous action spaces are policy gradient methods. Policy gradient methods attempt to directly update the policy parameters for a given problem. [132] Much of the initial work into policy gradient methods was focused on utilizing stochastic policies. This focus had two main sources. First, RL methods are reliant on adequately exploring the possible state-action space of the environment. Stochastic methods allow for greater exploration of this space even for deterministic environments, and may then lead to better solutions. Second, most early policy gradient methods were applied in an *on-policy* manner. That is, the learning occurred due to iteration while the policy was implemented. Without significant variance in the trajectories produced, little learning would occur. Introducing stochasticity at the policy level allowed for this exploration to happen naturally.

For much of the early work, it was commonly thought that the core foundation of policy gradient methods, the policy gradient theorem, may only be easily applicable to stochastic methods. This is due to the common form used, shown below, leading to vanishing gradients for deterministic policies. [132] This form was derived by looking at the log likelihood of executing arbitrary trajectories, τ, through the space while acting on a policy parameterized by θ.

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log p(\tau|\theta)\, \mathcal{R}(\theta)\right] \tag{2.3}$$

It can be shown that expanding the internal gradient in this case allows for a policy gradient without reference to the transition function:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{k\in\tau} \nabla_\theta \log \pi_\theta(a_k|s_k)\, \mathcal{R}(\theta)\right] \tag{2.4}$$

Clearly, a deterministic policy in this case would lead to a null gradient. This could be overcome by carrying a system model throughout the process, but this would defeat much of the benefit of using policy gradients in the first place. This led to a fear of using deterministic policies for these methods.

This fear was unfounded, as Silver et al showed that an analogous policy gradient can be calculated for deterministic methods. [106] This paper laid out the Deterministic Policy Gradient Theorem, shown in Equation (2.5), opening the door for deterministic policies to be implemented. The major trick used was to instead of using the return function directly, to the value function. In this way, gradient information could be incorporated into the returns directly, avoiding the vanishing gradient problem from the standard derivation. These included both simplistic on-policy and off-policy learning schemes. [106] also included some basic implementations which used this new gradient theorem. While a great step forward, these methods were susceptible to significant instability and required significant hyperparameter tuning to yield decent results. Many recent works have sought to improve these methods further.

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta \nabla_a Q(s,a)\,|_{a=\pi_\theta(s)}\right] \tag{2.5}$$

One direct attempt to improve these methods was Deep Deterministic Policy Gradients, or DDPG. [71] This method borrowed from the previously discussed Deep-Q-Learning approach, [78], to formulate an actor critic method to iteratively improve estimates of the policy gradient. Other key introductions included the use of a replay buffer to store and reuse past experiences, increasing the independence of samples used to train and update the

policy network, and the use of target networks, slowing the updates for the networks used to evaluate the quality of the policy and therefore stabilize the policy gradients calculated. These improvements increase the stability and usefulness of deterministic policy gradients, but also introduced a possible source of instability if the critic and actor policies began to diverge. This method was also very data intensive, requiring roughly a million experiences to be stored in the replay buffer and on the order of a million training episodes (each with potentially thousands of training steps...) for generalizable results.

Many other attempts fall under this same actor-critic idea. The general flows from the deterministic policy gradient theorem. [66, 37, 86] These methods explicitly separate the two terms within Equation (2.5). That is, the actor represents the policy, and the critic represents the value function. By iteratively improving the critic, better estimates for the eventual return of the current policy can be found. Using experience replay as was discussed in [71] and [78], these critics can actually be trained in a supervised manner. The actor can then be updated by following the policy gradient for each sampled experience. While this can lead to impressive results, and has been shown to be a fairly general approach to solving many continuous control problems [71], it is often an unstable process that requires significant hand tuning of parameters.

Another attempt to improve these methods was Asynchronous Advantage Actor Critic, or A3C. [76] As its name implies, A3C is an actor-critic method whose main contribution was the use of parallel agents contributing asynchronous updates to a shared policy. Each agent acts upon an independent environment such that, in theory, it will explore different subspaces of the state-action space than the other agents. As it explores its environment, it accumulates gradient information on how its current policy could be improved. The gradient information it has accumulated is applied to a shared model, for both the actor and critic networks, once the an individual agent's training episode is completed. The parameters of these shared networks are then synced back to the individual agent, and a new training episode is begun. In this way, individual agents communicate indirectly

through the shared networks, with the asynchronous updates ensuring each agent maintains a relatively independent parameterization.

A3C increased data efficiency over many actor critic algorithms. While the base algorithm used simplistic methods for training of the individual updates, other improvements, such as the inclusion of individual replay buffers and target networks, could be combined within this framework to further improve data efficiency. However, the parallelism that is the foundation of this method limits its applicability to real world problems. To illustrate this, consider the training of a robot arm. If this training were to be completed in the real world, a bank of robot arms would be required. Without this bank of arms, the method simply collapses to that of other actor-critic frameworks with target networks.

All of the algorithms considered above are considered model free approaches. That is, they do not explicitly attempt to identify the transition function of the environment they act in. Another class of policies that can be trained through reinforcement learning are model-based approaches. As the name would suggest, these do build an explicit model of this transition, and then attempt to optimize a policy based on this model.

Model-based approaches to reinforcement learning draw on a long line of model-based-control. This allows for some guarantees on performance and safety that are desirable for many systems. [6] Similarly, model-based approaches have shown to be more sample efficient than model-free methods [4] and have seen significant improvements in their performance recently. [91] However, they still have a significant source of uncertainty: the model. This requires very detailed analysis of the system to be considered, and can take significant effort and time. Janner et al discuss methods for bounding the modeling error based on a trust region approach. [55] While this method shows promise, it relies on a relatively restrictive formulation of the machine learning problem to formulate these guarantees. Polydoros et al describe an approach to learn a model online, allowing for adaptation from model errors. [90]

While model-based methods are certainly attractive approaches, this work will focus

on model-free approaches. Model-based approaches are currently sensitive to accuracy of their internal model, and therefore struggle with generalization. [55, 91] Similarly, model-free methods are often used as a part of model-based methods, so results for model-free methods will likely also hold for model-based methods as well. [112]

Throughout all of these modern approaches, one constant has been maintained. In order for a reasonable policy to be found the learning agent must explore the environment adequately. The next section will look at this more to understand how this can be done, and why it is necessary.

2.1.2 Exploration for Reinforcement Learning

The previous section outlined many recent developments in reinforcement learning. It identified model-free methods as a significant contributor to many of these breakthroughs. [71, 78, 76] It is expected that this will continue to be the case, but it is important to understand how these algorithms learn.

In his exploration of reinforcement learning, Sutton continuously comes back to the importance of exploration of the possible state action space. [112] the goal of exploration in a reinforcement learning framework is to identify new experiences that alter an agent's understanding of its environment. By finding scenarios that elicit this change in understanding, policies can be improved and new edge cases can be handled.

Exploration of the possible state action space has been an issue that has attracted interest for significant time in the reinforcement learning community. Early work by Thrun discussed what is meant by efficient exploration. [119] This included many ideas for possible exploration, including undirected and directed exploration. The key distinction between these two types of exploration is the application of randomness. Undirected exploration occurs through some random process, while directed exploration is more of a learned behavior that specifically seeks out new experiences.

These useful distinctions continue today, though many techniques are now a blend of

the two. That is, stochastic processes are used in target ways. Many modern reinforcement learning techniques explicitly follow some type of exploration. Two of the most popular classifications for exploration are action space exploration and parameter space exploration. [129]

Action space exploration is the most straightforward to explain. In short, some stochastic process is used to alter the output action from the policy, and this altered action is taken to generate the next experience. [112] This straightforward method allows additional stochasticity to be imparted on the system. The goal is this additional stochasticity will force the learning agent into new areas of the state space that wouldn't be seen under the current policy. This has been used widely in many of the recent frameworks for reinforcement learning. [71, 78, 76]

There are two major choices to be made when implementing an action space exploration policy. First, the random process that will be used. A common choice is simple uncorrelated white noise applied to the output action of the policy. Small alterations, such as different noise distributions are also fairly common. While useful, these policies don't necessarily lead to efficient uses of exploration. [119]

Other attempts at action space noise often use a stochastic process, such that the applied variance is correlated in time. One of the most popular is the Ornstein-Uhlenbeck process, characterized by the equation below:

$$dx = -\theta x dt + \sigma dw \tag{2.6}$$

Where x is the state of the process, w defines a standard Wiener process, and θ and σ are tuning parameters to determine the speed and allowable spread of the process. This produces a mean reverting process that, if x is applied to the actions taken by the policy directly, is centered around the policy output. This has been used extensively for continuous control problems, most notably in the original DDPG paper. [71]

A second choice in applying action space noise is whether this noise is applied adap-

tively through the learning process. This confronts one of the major tradeoffs when considering exploration of the state-action space: exploration vs. exploitation. One common approach is the ϵ-greedy approach. This approach simply uses a greedy action, that defined by the current policy, with probability $1 - \epsilon$. In cases where the greedy action isn't taken, the randomized action is taken. This has been applied to discrete spaces in many major frameworks, such as Deep-Q networks in [77] and [78]. This can also be applied in continuous environments, such as in [71]. Other options include slowly reducing the noise parameters such that the policy being trained on continuously approaches the policy being trained.

Parameter space exploration takes a different tactic. [89] As the name suggests, this adds noisy exploration to the parameters of a policy instead of the action output. This has been compared to zeroth-order optimization methods, such as evolutionary training. [28, 104] The idea is that this will more efficiently search actions in a nearby space, allowing for the policy gradient to be more accurately estimated. Recent works have shown this to have positive results for many of the modern model-free approaches to reinforcement learning. [71]

In all of these approaches though, primary goal is to push the learning agent away from the state space that is elicited under the current policy. This allows for local minima to be avoided, and high quality behaviors to be identified from scratch. This represents a major issue though, as one of the goals of reinforcement learning has been to identify safe policies when previous examples don't exist. The necessity for exploration to find these runs counter to this goal. This is part of why so many reinforcement learning breakthroughs have been made in virtual domains. [64] However, modern modeling and simulation technology has led to more and more realistic environments to be developed for training of systems that will be employed in the real world. [88] These attempts at training in simulated environments have their own issues though, as transferring policies from one environment to another is still a significant challenge for reinforcement learning based systems. [64] The

next section, Section 2.2 will discuss this issue and current approaches in more detail.

2.1.3 Summary

This section discussed some of the breakthroughs for autonomous systems that have come as a result of modern reinforcement learning frameworks. These include superhuman levels of performance in many games, and the control of mobile robots. [20, 64, 71, 78, 61] A common theme across these approaches is the need for exploration of the state/action space. The groundbreaking Go playing system, AlphaGo Zero, used over 4.9 million games during it's training process. [107] Without adequate exploration, the resulting policies run the risk of overfitting to a small, non-representative subspace that can cause significant issues if the system is ever forced out of this known space.

For embodied systems, such as robotics or autonomous vehicles, that don't have exact virtual representations this causes a major problem. Gathering real data for a more full picture of the state-action space is expensive and would result in possibly unsafe conditions. To combat this, it is common to turn to simulation, generating additional training data in a virtual environment. This is a key point, as nearly every reinforcement learning trained policy will require some initial simulation based training.

While this sounds like a panacea for the problem, training in simulation brings about its own issues. Kober discussed some of these as so-called *undermodeling*, and noted that they often cause issues with respect to transferring policies from a simulation based training environment to the true system. [64] While this undermodeling is one of the reasons why it is difficult to transfer policies from simulation to the real world, it is certainly not the only reason. This issue, the difficulty of transferring policies from the simulated world to the real world, has spawned a whole field of study in the *Sim-to-Real* area of robotics research. The next section will discuss current literature from this area and identify gaps that remain.

2.2 Sim-to-real Approaches

The previous section discussed some of the recent breakthroughs with regards to modern reinforcement learning frameworks for development of control of autonomous systems. While these techniques have certainly produced impressive results, they are often limited to solving problems in the virtual domain. There are multiple reasons for this, but one of the major reasons is the need for wide ranging exploration during policy development to be successful. This is much easier to accomplish for a virtual system. This is because exploration in the real world may take a prohibitively long time in the real world, where all interactions must happen in real time. Similarly, this exploration may push a real system into dangerous areas of the state action space.

As an example of this, consider an autonomous car. It would be a disaster to attempt to learn any driving behaviors from scratch in this sort of environment. First, the car would likely have to go through many crashes before learning proper application of breaks and steering. Additionally, even if bounds on the behavior could be applied that could guarantee safe outcomes, learning would occur very slowly. Every new situation would need to be encountered to ensure the autonomous agent had the experience necessary to overcome possible perturbations such as changing road conditions, the local nature of driving laws, and the average dose of road rage encountered from other drivers.

To address this need for exploration, early phases of autonomous policy development utilize simulation environments. These methods attempt to model the most salient features of the real world so a learning agent can get a reasonable approximation of a good behavior before being used in the real world. While promising, this has faced many issues of its own. Mainly, the policies trained in the simulated environment fail to work appropriately once transferred to the real world. This is often attributed to the so-called *reality gap*, the inherent differences between simplified simulation models and the real world. This has spawned an entire field of research called *Sim-to-Real*. This section will cover some of the major

approaches for sim-to-real from the literature. This will be broken into two major sections: simulation development and transfer learning. These can be seen as two dimensions of the sim-to-real problem. That is, one factor is minimizing the reality gap between the simulator and the real world through intelligent design of models, while another factor is developing algorithms that can be more robust to this changes in the environment.

2.2.1 Simulation Development

One of the oldest and most commonly used approaches to the sim-to-real problem is to use ever high fidelity simulations. The idea behind these approaches is relatively simple: if the reality gap is the main source of lack of transference, minimizing this gap will lead to better transference of policies. This has led to the development of numerous simulation environments, ranging from general purpose physics simulations to highly specific simulations of a single system. [2, 65, 101, 103, 121] Clearly it would be nearly impossible to discuss all such efforts and simulators individually. However, some major recent contributions and general trends from the literature will be discussed here.

Zhang et al describe an early attempt at developing a simulation for training a robotic arm pointing task in simulation. [139] In this work, a custom simulation environment for a 2D grasping task was developed. A 3-DOF robotic arm was modeled, with simple visual outputs for an end-to-end policy for reaching a goal point with the end effector. The behavior was trained with a Deep Q-Network. [78] The simulation used multiple scenarios to train the behavior policy, including randomizing the location of the root of the robot, the link lengths of the robotic arm, and adding noise to the output image. This can be seen as a type of domain randomization, [120] which will be discussed in more detail in the next section. While the policy performed reasonably well in the simulation environment, it failed to transfer to the true environment under any conditions.

James and Johns describe another approach to developing Deep Q-Learning based policies for a 7-DOF robot arm grasping problem. [53] In this work, they develop a custom

high fidelity simulator to perform a grasping task. They discuss an iterative approach to simulation development, where first a rudimentary simulation was developed for a proof of concept. This allowed for successful training in simulation, but the policy again failed to transfer to the real world. Further effort went into increasing the fidelity of the simulation, including a more accurate contouring of the robotic arm and additional surface texture modeling. This led to a semi-successful transfer to the real world. That is, the learned behavior would be considered at the pointing task, where it consistently made contact with the object, but failed to grasp the object.

Given these two examples, there is clearly some level of fidelity required for policies to be transferred successfully. While both of these cases used pure 2D cameras for development of visual input to the system, it is possible to gain additional data from the system by using depth sensors in tandem with a visual camera. Planche et al took this approach, using a 2.5D visual representation to solve the problem. [88] In doing so, they developed a high fidelity model of the scenario in CAD to allow for a realistic return of the 2.5D sensor. This included a detailed analysis of the noisy returns from the real sensor to calculate a realistic texturing profile for the returned images from the simulation environment. They showed that object detectors and pose estimators trained solely on these synthetic images outperformed similar detectors trained on a smaller set of low quality images from the real world.

This presents a strong case that simulation based training can yield significant returns for autonomous systems. While this wasn't a reinforcement learning trained system, it is likely to have faced many of the same challenges. However, this was a very intensive process requiring many hand tuned noise features for the depth images. Additionally, no evaluation of which additional modeled phenomena, such as background blurring, stereo-matching, or pattern matching, was most significant in allowing for this transference. As such, it's possible that a simpler process may have led to similar transferred performance.

While not directly applicable, other recent breakthroughs may allow for even further

increases in simulation fidelity. Generative Adversarial Networks, or GANs, have seen increased use in a wide range of fields. [30] This technique uses two networks, a so-called generator and a discriminator, to train against each other in an adversarial manner. That is, the generator network produces some state vector, often an image, from a target domain. The discriminator then attempts to identify whether any given state vector is from the true target domain or was produced by the generator. By iteratively training these networks against each other, it is possible to develop a generator network that is good enough to fool even human observers. [58]

While GANs are most often thought of for producing one-off images or videos, they are also beginning to see use in other areas. Work has been done to use a GAN to represent behavior in autonomous vehicle simulations. [68] Similar ideas have been proposed to augment data generated by either real data for transitioning between different task domains. [124] It has also been shown that applying GANs can be integrated with simulated data to produce highly realistic synthetic data. [8] This led to an impressive reduction in necessary real world data required to train the policy.

However, the key weakness in all of these approaches using GANs is the significant amount of data required before to train the generator in the first place. Each of the above applications used a static platform, such that collection of training data for the training of the GAN was straightforward. For mobile platforms there may be important couplings between data policy implementation and data acquisition. Consider a policy that leads to jerky behavior, leading to lower quality sensor data. If this data isn't captured during the training of the GAN, it is unlikely to be accounted for and may lead to a lack of transference.

Clearly, the use of simulations for training autonomous systems have begun to be successful. However, this is still further work in identifying why one simulation seems to work, while another fails. The next section will look at a slightly different approach to the sim-to-real problem. Instead of building better simulators, it may be possible to build better algorithms or better training procedures. One way to approach this is transfer learning, and

will be discussed next.

2.2.2 Transfer Learning

While high fidelity simulations have played an important role in addressing the reality gap, another area of research in the reinforcement learning community, as well as the broader machine learning community, is transfer learning. The broad goal of this research area is to develop techniques that allow for algorithms that were trained in one scenario to be applied to another scenario. This can include transferring from one environment to another, but also includes things such as transferring knowledge learned on one task to another. [117] There are many approaches to this, including learning hierarchical approaches to a given problem such that different components can be combined in a modular fashion to solve new problems. Clearly, there is a broad range of techniques in this area, so this section will focus on those that have two salient features. First, they will be focused on embodied tasks, where the transfer goal is to move from a simulated environment to a real system. Second, this will focus on works whose primary goal is the development of autonomous agents trained through reinforcement learning.

Taylor provides a useful survey of many of the possible methods for transfer learning and the different cases it can be applied. [117] This includes adjusting the state space of the learning agent, adjusting the action space of the learning agent, or adjusting the environments transition or reward functions. It also describes what knowledge is being transferred to improve the final agent. For the sim-to-real problem, we are considering a case where the state action space is consistent between source and target environments, but that the transition function, the system being modeled, is allowed to change. Similarly, the transition is accomplished by bringing the policy from the source environment to the target environment. Depending on the behavioral algorithm used, the value function approximation may also be transferred between the environments.

One approach to actually accomplish transfer learning is to use many environments for

training. Cutler et al describe just such an approach that combines transfer learning with progressively higher simulation environments, such as those discussed in the previous section, called MFRL. [21] This builds on much earlier work in transfer learning that showed learning on progressively more difficult tasks can lead to more efficient learning. [102] This is accomplished using multi-fidelity simulations to train reinforcement learning algorithms that can be applied to the real world. This work draws on recent work in the area of multifidelity optimization to attempt to not only train in different simulation environments, but to select which environment to train on at any given timestep. This is done by training separate agents on each simulated environment.

The agent for the lowest fidelity environment is trained until it is confident in its abilities. Knowledge from this agent is then transferred to the next agent, used to learn in the next progressively higher fidelity environment. If an agent ever becomes confused or begins performing poorly at its current fidelity level, it can provide its newly learned parameters to a lower fidelity agent for training on a lower fidelity level simulator. This passing back and forth of parameters allows information gained at different fidelity levels to be used by each agent. This follows the concepts of transfer learning to iteratively improve policy performance by training on increasingly complex versions of a problem. This method also uses periodic real world sampling to correct inconsistencies learned through simulation based training.

While the results of this method are certainly impressive, there are a few shortcomings. First, it assumes a strict ordering of fidelity between the models used during training. That is, it assumes the ordering of these models is known a priori, specifically by enforcing an assumption that the parameters of a higher fidelity model are a superset of the parameters of each of its associated lower fidelity models. If there isn't this strict superset relationship, or the relative fidelity of the models is not known a prior, it is likely the policy parameter transfer between agents may fail. Similarly, while it uses a multifidelity approach, it does not comment on how these models should be constructed. It merely assumes that the high-

est fidelity model has sufficient fidelity to produce transferable policies on its own. Finally, while efficient in its use of real world training samples, this necessary interaction with the real system limits this approach to problems where use of the real system during training is feasible.

Another form of transfer learning uses what is called domain adaptation. [130] In this form of learning, the goal is not so much to learn a policy that is transferable between a source and target environment, but to learn a mapping between the two environments. In this way, data from the source environment can be transformed such that it mimics data from the target environment. In this way, policies can be trained on the adapted data and then applied directly to data after transfer. This has seen significant use in areas such as computer vision and classification, but has not seen significant use in control systems or for reinforcement learning specifically. [9, 130] Zhang et al discuss a similar approach for an embodied grasping task that uses an adversarial approach to domain adaptation. [138]

Another recently developed approach for handling the sim-to-real transference problem is called *domain randomization*. [120] The general idea behind domain randomization runs counter to that in the previous section, where the goal of simulation is to reduce the reality gap as much as possible. Instead, domain randomization poses to randomize certain parameters in the simulated environment, forcing the learning algorithm to produce a policy that is robust to perturbations from the expected environment. This has shown impressive results, leading to efficient uses of real-world experiences and even some successful zero-shot transferences where no real world data was used. [15, 82, 74, 120]

What can be randomized is generally left open to the designer of the system. For vision based systems, randomizing colors, textures, and lighting conditions has been shown to achieve positive results. [120, 82] For control of physical systems, changing the dynamics parameters of the system, such as the strength and orientation of gravity, friction, and other miscellaneous phenomena has shown good results. [15, 82, 74]

These impressive results are an exciting new area of research, and can be combined with

previous methods of incremental transfer to produce striking results. [82] However, the implementation of what should be randomized within these methods is an open question. If the randomization bounds are left too loose, the task may become too difficult to learn and no policy is adequately found. The alternative may also happen, where without sufficient randomization the policy fails to be robust to the changes when transferring between the simulated and real worlds.

While it has been proposed that these randomization parameters can be learned, it may be more useful to evaluate the models themselves to identify which phenomena have the greatest impacts on behavior. This is still a notable gap, where the features to randomize are largely identified in an ad-hoc manner. It may be useful to instead identify the relative importance of different phenomena directly prior to applying domain randomization.

James et al propose a technique that combines some of the possibilities of domain adaptation and domain randomization. [54] This technique defines a mapping from randomized simulations to a canonical simulation. In this way, the behavior can be trained directly on the canonical simulation while the mapping can be made robust through advances in domain randomization. This was shown to have better results that pure domain randomization, but still was susceptible to artifacts from the mapping process.

A common need for all of these approaches is a reasonable starting simulation. While in theory, these techniques can be applied to "low-fidelity" simulations, the examples given are used on fairly intense simulation environments that take significant effort to develop for a new system. Additionally, many of the most promising works coming from domain randomization require the identification of critical phenomena to alter to find these robust policies. While techniques such as those proposed in [74] and [82] and others show this may be possible to incorporate directly in the learning process, this may be discarding valuable information away that can be taken from the modeling environments themselves.

2.2.3 Summary

The need for exploration in the development of reinforcement learning based policy development has led to the use of simulation based training. While this addresses many of the issues related with policy exploration in the real world, transferring policies form simulations presents its own problems. [53]

Of these, undermodeling of a system is a likely culprit in many of these situations. [64] This is supported by the ability of high fidelity simulations to produce reasonable transference results. [8, 88, 124] However, the use of high fidelity simulation doesn't always lead directly to transferable policies, and is largely conducted as an ad-hoc approach. One possible reason for this is the common end-to-end approach to learning. This hides important features in the simulation model that may have an outsize impact on successful training. This leaves a gap in our understanding of the impacts of modeling choices on transfer of learned behaviors.

Other approaches to the sim-to-real problem often follow from transfer learning. The goal is to setup both the learning algorithm and the environment to find solutions that are robust to changes in the environment. [120] This allows for an indirect solution to the undermodeling problem. While this has worked in some cases, it is also a largely ad-hoc approach. There is little understanding of how to identify which phenomena in the simulation should be randomized, and which are most important to develop robustness when transferring policies. More worryingly, this approach often creates additional difficulties in training the original policy. [74] This can create scenarios where overly conservative policies are found, or the problem becomes too difficult to solve at all. This difficulty in identifying which phenomena a policy should be robust to mirrors the gap from fidelity based approaches, where there is a lack of understanding with respect to what portions of a system are necessary to capture in the simulation.

To try to get a better understanding of this, the next section will discuss results from the broader modeling and simulation community. The hope is that this search will identify

methods to intelligently find the portions of a simulation that are most import for transference.

2.3 Modeling and Simulation

The previous section described approaches to simulation design specifically for training autonomous systems taken from the Sim-to-Real research area. One of the major philosophies that has come from this research is the attempt at designing models of sufficient fidelity for transference to occur. It was shown that for sufficiently high-fidelity simulations, reasonable levels of transference could occur. [88] However, no matter how complex, how detailed, or how calibrated to real data, a model will remain a simplification of the real world and so this reality gap will remain. [96]

In the modeling community, there is a common saying that "All models are wrong, but some are useful." This colloquialism gets at a deep understanding of the use of models in many fields: they are inherently simplifications of the real world meant to evaluate the effects of some idea. This implies that the data produced by a model will always have some error when compared with data from the real world. However, depending on the nature of that error, the data may still be used to identify trends, predict behaviors, or evaluate options in a useful way. Part of the goals of modeling as a practice is to reduce this error to increase the range of useful data that can be produced. This is especially relevant to reinforcement learning, but lessons can be taken from the modeling approaches of other disciplines. This section will now consider approaches to modeling and simulation from other research areas. First, it will look at some of the theoretical background between what the goals of modeling and simulation are and how they can be accomplished. Next, techniques for model development in general will be discussed from a range of disciplines. Observations from these areas will be combined with those from the two previous sections to formulate the specific gaps to be addressed by this work.

2.3.1 Modeling and Simulation Theory

Throughout time, models have been used by people to better understand the world. Inherently, a model is a simplification of the real world.

Before fully discussing the ways a model for a simulation of a system should be developed, there needs to be some discussion of what a model actually is. No matter how complex, how detailed, or how calibrated to real data, a model will remain a simplification of the real world. [96] This generalized view of a model is found throughout the literature on modeling and simulation. [69, 87, 100, 137] The purpose of this simplification may be varied, and the forms a model can take mirror this. From simple algebraic equations, interacting agents following fuzzy sets of rules, to physical representations like wind tunnel models, models can show significant variety and in both form and use. During early phases of the design process modern engineering practices heavily rely on the ability of executable models for analysis and prediction of system behavior.

George Box got at a similar idea with his famous quote: “All models are wrong, but some are useful.” [10] This colloquialism gets at a deep understanding of the use of models in many fields: they are inherently simplifications of the real world meant to evaluate the effects of some idea. This implies that the data produced by a model will always have some error when compared with data from the real world. However, depending on the nature of that error, the data may still be used to identify trends, predict behaviors, or evaluate options in a useful way. Part of the goals of modeling as a practice is to reduce this error to increase the range of useful data that can be produced. A common way to discuss this is through the lens of ***fidelity***.

While fidelity has a common intuitive meaning, it is hard to pin down a precise definition that is useful for its analysis. When discussing a model, high fidelity is good and low fidelity is bad, witch caveats for if computational expense is taken into account. As defined in the Oxford English Dictionary, fidelity refers to: ”the quality of being faithful; strict conformity to truth or fact.” [25] This definitions is useful in understanding what is gener-

ally meant, but also vague and difficult to apply to the context of modeling and simulation. What does it mean for a model to be faithful? Is a model that produces quantitatively closer results but qualitatively worse behavior more or less faithful than a model that produces the reverse? Would a model with that has more details of lower quality be consider higher fidelity than a model with fewer details that are more accurate?

Many have tried to get a better definition in the context of modeling. This can be seen in the countless efforts to define and understand fidelity throughout time. [60, 33, 42] This continued to such a point that the Simulation Interoperability Standards Organization (SISO) created a working group to study fidelity within the context of simulation known as the Fidelity Definition and Metrics Implementation Study Group (ISG-FDM). This group produced a final report on its findings, outlining the current state of art in defining fidelity and how it can be used to improve simulation interoperability. [34] The main findings from this report included the definition of related concepts of fidelity in a glossary of terms, an initial definition meant to be used in practical modeling and simulation settings, metrics to quantify and understand fidelity, and suggestions for future research in the area.

The reactions to this report were mixed. While the concepts identified within the report were positively received, the metrics identified were lacking. [83, 99, 100] A second ISG commissioned by SISO, the Fidelity Experimentation Group (ISG-FEX) was tasked with implementing and experimenting with the fidelity framework outlined. Again, this group produced a report on using the standards set out in the ISG-FDM report. [99] The main conclusions drawn from this report were the limitations of the framework in actually implementing a fidelity standard that could be useful in the modeling and simulation community. Recommendations from the report include the importance of defining and identifying relevant information about the portion of reality that will be simulated, known as the referent, and a distinction between model fidelity and simulation fidelity. The difference is due to the definitions of *modeling* and *simulation* in the ISG-FDM report. That is, modeling is "a physical, mathematical, or otherwise logical abstract representation of a system, entity,

phenomenon, or process with its own assumptions, limitations, and approximations" while a simulation is "a method, software framework or system for implementing one or more models in the proper order to determine how key properties of the original may change over time." [34] Additionally, a link was drawn between fidelity and the validation of simulations, though a distinction between the two concepts was drawn.

Among the members of the initial ISG working groups on fidelity, Roza attempts to lay out a more rigorous definition of fidelity as well as methods to measure it. [100] Through this work, a few main concepts for fidelity were identified. Namely, accuracy of behavior and phenomena representation are considered. These are useful to understand, as this is a common approach taken when attempting to increase the usefulness of a simulation model.

As previously discussed, a model is an inherent abstraction from the real world. [137] The level of abstraction can be understood through the concept of fidelity. Fidelity can be seen at many levels in a simulation, ranging from fidelity of the simulation as a whole to the fidelity of the component models that make up the simulation. Clearly, the fidelity of low-level component models will impact the fidelity of the total simulation, but this relationship is just beginning to be understood.

Many of the methods for discussing fidelity in the literature rely on qualitative comparisons, such as those discussed in the use of functional fidelity above. [45, 49, 126] In general, these attempts to define and compare fidelity look at the abstraction choices made in models directly. While this approach is useful in many fields, it is likely to become prohibitive when many component models are interacting. This also creates inherent subjectivity in the evaluation of fidelity, as it is reliant on expert opinions to identify the abstractions made in the first place. The flight simulator field has produced many measures of fidelity that attempt to move past these qualitative descriptions though, and a selection of these descriptors is discussed below.

The FAA takes a multi-criteria based approach to fidelity classification of airplane flight simulators. [1] These criteria span multiple categories, from visual and auditory cues to

predictive modeling of the aircraft, such as aerodynamics and avionics models. Depending on various thresholds of acceptance for these many evaluation criteria, a categorical fidelity designation is assigned that signifies the simulators acceptable use for certification and training of pilots. While this provides a possible framing for evaluating simulation models for the design of autonomous systems of systems, it is only applicable to the entire macro-level simulation and attempts to decide the acceptability of the simulation in a near binary fashion. Individual criteria could be used to identify modeling weaknesses, but the complexity of the evaluation may hinder the ability to rapidly evaluate and improve simulations.

Burnett proposed a simplified version of these measures to evaluate fidelity of flight simulation trainers. [12] This scale assigns a 0 to 10 grade depending on the level of abstraction a model implements with respect to the real aircraft. A rating of 0 is the actual system being simulated, with greater levels implying greater abstraction. A level 10 model is considered a static pictorial representation of the system. While the simplification of evaluation is useful, this may result in incomparable results. Additionally, the collapse of fidelity into a single categorical value is likely to miss the influence of specific details of a model. this simplicity is attractive though.

For this work, fidelity of a model will be considered simply as comparison between the number of phenomena represented in some referent for a system and those captured in the model itself. While this won't allow for direct comparisons, it gets at many of the ideas expressed above. That is, as fidelity increases, the distance between the truth and the model should decrease. Similarly, it allows for simple comparisons that should follow the intuitive relation between fidelity and transference.

Given a reasonable idea of what fidelity is for a model, we can now talk about their use. The practice of building models and executing them in a given environment is called simulation. [137] Simulation tools allow for virtual experimentation to augment the information that is available regarding a system's behavior during the design process. This allows for

imaginary systems that are still being conceptualized to be probed, design alterations to be considered and contrasted, and experiments that may be too dangerous or cost prohibitive to be conducted.

Systems theory provides much of the theoretical foundations when discussing modeling and simulation. Systems theory has a long history and attempts to describe general principles about systems that go beyond disciplinary analysis. [75] Systems theory considers two main aspects of the system under consideration: its behavior and structure. The behavior of a system is defined as the relation between time histories of input and output variables. The structure of a system can be taken as the internal behavior of a system: the internal states and their transition functions that lead to the external facing input output relationships. One of the greatest powers of systems theory is its property of closure under composition, also known as closure under coupling. [75] In short, a portion of systems theory is the decomposition of macro-level systems into their micro-level components. A counter to decomposition is composition, or the combination of systems to produce a new system. This hierarchical compositional process has been shown to remain consistent with the base of systems theory, so objects built through the coupling of independent systems can also be described and analyzed using the original systems theory foundations. Zeigler et al discuss various methods of formalizing systems theory for modeling and simulation. [137] The three major formalisms considered are differential equation system specification (DESS), discrete time system specification (DTSS), and discrete event system specification (DEVS).

DESS descriptions use interacting differential equations of system properties to describe their structure and behavior in a continuous time base. They are some of the most fundamental descriptions of systems, and are often the first systems considered when discussing systems theory. They are often used in scientific studies, and are the foundation for much of our understanding of the physical world. One of the greatest advantages of these specifications is for special cases of these specifications, analytical solutions to system be-

havior can be found. However, these cases are relatively rare and often do not represent systems found in the real world. For solutions that cannot be reached analytically, numerical integration methods are required, and can be costly. Finally, many phenomena do not have known differntial equation based forms. [137] DTSS descriptions utilize a similar structure while relying on a discrete time base for the description of system behavior. They may be either fixed or variable time interval descriptions. [137]

DEVS descriptions take a much different approach, treating time as a dependent variable and instead attempting to define events in the system that trigger a change in the description of either a systems structure or behavior. Attractive properties of DEVS include a compatibility with closure under coupling, and provable universality of descriptive capability. [137] In short, DEVS based descriptions can simulate any other system description to an arbitrary closeness. DEVS has seen success in many realms, and has even been used as a basis for initial work in formalizing and verifying emergent behaviors, as shown by Kiriakidis. [63] However, in this case the control of the micro-systems was done in a supervisory and centralized manner. Many emergent systems are due to the decentralization of control where this formalization would not be applicable.

While powerful in their theoretical capabilities, DEVS based systems have somewhat fallen out of favor for analyzing complex systems with agent-based modeling strategies often replacing them. [105] Agent-based models (ABM) are a common approach to modeling complex systems of interacting components, and have found success modeling systems from disciplines as diverse as ecology, sociology, economics, and air traffic management. [7, 32, 41, 73] They are well known to be useful in studying and analyzing systems that may display emergence. [14, 47] While ABM has attractive properties and is a natural approach to modeling emergent systems, it lacks much of the formalism of other modeling frameworks. [22, 79, 133]

Much of this theory and these frameworks are useful in evaluating high-level choices in simulation design.However, they don't seriously touch on the development of the models

used within them. To start along this goal of looking at lower level modeling decisions it is important to return to that original point: models are a simplification of the real world that can be represented as a collection of phenomena. Understanding this, model development and improvement can be seen as a selection problem, identifying the most important phenomena to capture within the model. The next section will now take a more practical approach to this problem, looking at methods for model development from a range of disciplines.

2.3.2 Simulation Model Development

The previous section discussed some of the theoretical background on what modeling and simulation attempts to accomplish. Understanding that a model is really a collection of phenomena that represents a simplification of the true system of interest is key in understanding why some models work well and others fail. This section will now discuss some of the relevant literature that use this framing to then select or develop models appropriately.

One common area in the literature for model development focuses on modeling composition. [27] While promising, even recent techniques from this area are largely focused on attempting to build a model from existing components. [29, 39] That is, they compose a modular model of a system from many individual component models by checking input/output interfaces with little regard to validity of the final model. This doesn't necessarily allow for the comparison of different models for each component, and does little to verify the accuracy of the full model. These techniques often are used as a starting point to further refine a model, as they may lead to undesirable emergent behaviors not necessarily seen in the true system. [128]

Understanding the effects and sources of emergent behavior may be a useful analog to understanding modeling effects on reinforcement learning, as learned behavior is often considered an emergent property. [46]. The existence of effects due to modeling choices is supported by much of the existing literature on emergence, but further research is required

to develop more than simple qualitative descriptions of these effects. Varas briefly discusses the effects of modeling choices, such as homogeneous agents compared with noisy agents in the emergence of traffic patterns in city blocks. [128] However, this effect isn't probed further, with other potential simplifications not considered. Tolk has also discussed the possible limitations on evaluating and understanding autonomous behaviors through simulation due to epistemological constraints. [122] However, these results have not yet led to obvious methods for determining allowable simplifications in modeling choices or the identifications of simplifications that significantly impact the macro level behavior of autonomous systems.

Szabo and Teo have attempted to address the problem of emergence with linguistic and grammar based approaches to the verification of emergence in complex systems. [113, 118] These approaches are promising, and are useful in determining the root causes and quantifying the level of emergence that is seen in macro-level behavior. [113, 115] However, by accounting for the complexity of system of systems behavior, they have fallen back to intractable levels of work for verification for large systems. [114] They are also limited to verification of whether a property or behavior, once identified, is actually emergent or can be reduced to explanations stemming from micro-level behavior. While important, this does not verify the accuracy of the behavior representation itself and has limited use with regards to reinforcement learning based behaviors. These works with emergence do show that understanding of complex behavior through modeling and simulation is a useful tool, though.

Some tangential work by Hunter et al was discussed previously, where the authors investigated so called "functional fidelity" of a model. [49] While this work was focused on functional models of a system, some of the conclusions may be applicable to executable models used in reinforcement learning training as well, with a bit of rework. Namely, the possible functionality space within a functional model can be considered an analogue of the possible behavioral space of a simulation model. This suggests that it would be possible to

compare models based on the behaviors that are presented during simulation.

Other work in verification and validation of complex system behavior has either followed an ad-hoc approach or been limited in its applicability due to the combinatorial explosion of the state space as more agents are added to these complex systems. [79] A promising approach to verification of complex systems is compositional verification. This method is a general framework for the verification of complex software systems that focuses on bottom up verification of components as they are built into an application. While promising, current methods used in compositional verification are often based on either formal proofs of correctness that are limited in their general applicability and scalability, or on model-checking through testing and debugging, which may allow for complex behavior to emerge through edge cases that have not been considered. [79, 94] This limits the ability of current methods in their applicability to verification for modern behavior algorithms that are based on black box neural networks. However, they do identify the importance of looking at different levels of the model, considering the importance of different component models. This can be seen as similar to the idea of identifying the importance of different phenomena within a more holistic model.

In looking at model selection for engineering design, Radhakrishnan and McAdams describe a method for based on utility theory. [93] This is formed by evaluating the utility of a set of proposed models for a given problem. This utility is based on evaluating the truthfulness of each model. In essence, this is evaluating the relative validity of each model. In this case, the "most truthful" model must be known a prior so each other model may be judged in a relative sense. Clearly this may not be possible in all cases, especially in the world of autonomous systems, where small inconsistencies in otherwise high fidelity models can be exploited erroneously. However, this method does illustrate the need to use a relative comparison between many models to identify strengths and weaknesses in different models. It also points out the need to consider high level goals for the problem at hand when evaluating the utility of each of the possible models.

Panchal et al describe a value of information based approach to model refinement in [84]. In this context, value of information refers to a quantitative measure of the possible potential improvements that can be gained from further tuning of a model. In this case, it can be measured as the improvement of a design decision based on the additional information gained by further refinement. To estimate this potential, this requires having accurate bounds on the potential performance metrics as predicted by the simulation. For autonomous systems, these bounds are nearly impossible to predict in a non-trivial way. That is part of the reason why transferring behaviors from simulation to the real world is so difficult in the first place. However, this work spoke to an important point when evaluating simulation models. That is, a model should be evaluated in the context of the decisions it produces.

Some of the original analogies for understand machine learning broadly and reinforcement learning specifically are the training processes used for human learning. While these analogies are certainly not perfect, they can shed some light on possible approaches for autonomous systems. One of the areas with the greatest success for simulation based training is pilot training. Flight simulation is a critical component of the training and certification process for many commercial pilots. [3, 1] High fidelity simulators are a critical component in much of this and have been studied extensively in the context of human factors.

An important result from the human factors field shows that fidelity of a simulator does not perfectly correlate with learning transfer and may even cause worse performance when compared with lower fidelity simulation. [42, 43, 60, 69] This leads to an understanding of fidelity as a complex relation. This could stem from poorly defined measures of fidelity, or a lack of identification of the most relevant portions of fidelity to be considered. For example, fidelity is often treated as a one dimensional quantity in the literature , but this is likely to miss or confound multiple important concepts that impact the ability of a model to properly represent varying scenarios. [34] This leads to the possibility of a scalar metrics reporting high fidelity overall when the dimension of fidelity with greatest importance would rate as

low fidelity.

A specific example of this can be seen in the work of Winter, Dodou, and Mulder. [134]. In this, the authors performed a meta-analysis looking at the effects of including motion cueing in the simulation environment during training on the transference of lessons learned in training to actual flight. The results were interesting, showing that the increased fidelity was most useful for novice trainees, but actually led to a decrease in transference for expert trainees. This may be because expert trainees were able to detect subtle changes in the behavior of the motion system that were distracting. Hays and Singer noted similar issues in earlier work, noting that special consideration should be given to specific phenomena as they will have an outsize impact on actual transference of training. [43]

Yip studied the impacts of model fidelity on the results of a optimal control design for an industrial boiler. [135] In this study, two control architectures were considered: model free and model predictive control. In general, the study found model predictive control to be yield better boiler control strategies with respect to energy used and ability to track a target temperature. However, low fidelity models used could produce worse results than a model free approach, leading to the suggestion that higher fidelity models be used when possible. This analysis did not take the development effort necessary to produce these higher fidelity models into account though, and again used a fairly simplistic definition for fidelity.

The difficulties in defining fidelity rigorously were previously detailed in Section 2.3.1. To avoid some of this complication, this work will use fidelity similar in idea to that defined in [100], where it considers the phenomena used to develop a model. While this is somewhat simplistic, and therefore cannot be used alone for direct comparisons of models, it does follow from the intuitive meaning.

Turner and Mavris have developed a framework for iterative development and validation of conceptual mission models used in agent based modeling using weighted decomposition. [123] The method can be briefly described as follows. First the mission objective

is defined and decomposed in a hierarchical fashion. This results in the identification of a hierarchy of impact variables (IVs) that contribute to the overall mission, and the relationships between these variables (IVRs). These IVRs can be viewed as transfer functions, similar to those seen in Holland's description of constrained generating procedures in his description of modeling emergence. [46] The IVRs are then used as indices in a matrix, called the Subjective Impact Matrix (SIM) which can be used to assign weights to these relations to provide an initial estimate of their impacts on mission performance. Using the weights assigned in the SIM, the fidelity of various models that impact these relations in simulation can be defined with the help of a morphological matrix. Based on the fidelity requirements identified, the models are developed and imiplemented in a simulation environment. A sensitivity analysis of the parameters of these models can then be conducted to yield an Objective Impact Matrix (OIM). Comparing the resulting OIM with the initial SIM can then yield evidence supporting either the validity of the simulation environment if they agree, or the need for further refinement of the micro-level models. This refinement can be done iteratively by replacing the original SIM with the newly defined OIM.

While this framework is promising in its capabilities, it lacks in certain areas. First, this framework may be highly sensitive to initial weights assigned in the original SIM. Low initial weighting will yield low fidelity modeling of these relations, which may miss their impacts on system behavior. The authors note this, as the vehicles simulated showed good agreement when aggregate metrics were used, but showed significant variance in agent level behavior with discrete behavioral tendencies. [123] Additionally, this framework uses a simplistic view of fidelity. Models are assigned to one of three fidelity categories: Low, Medium, or High. This simplification to a single lumped qualifier may miss the varied impacts phenomena choices made while constructing individual models. More rigorous investigations of these impacts may be useful in further refining models.

Other fields also have useful techniques for model selection in developing simulations. Matchmaking technologies in web services attempt to find appropriate services given a set

of requirements defined by a request and mirror the attempt to find appropriate simulation models. Common approaches to this are input-output matching, which has seen some use in automated simulation design under model-based systems engineering. [23, 136] While useful in limiting the models to be considered, these methods do not consider internal behavior of the service, or model, and yield relatively coarse results. Semantic based matching methods may yield better results, but would need to be extended for their use in simulation model selection. [36]

Many other disciplines have also looked at the issue of model selection. These techniques are sometimes called *model discrimination* and often have significant theoretical backing. [5, 44, 48, 80] For example, political studies proposes many methods for discriminating between behavioral description models. [16, 17] While these techniques are certainly powerful, they are often tied to simple analytical models. These models have negligible evaluation costs that allow for extensive experimentation to determine areas of similarity and dissimilarity. While this can also be the case when looking at low-level behaviors of autonomous systems, this is certainly not the case when considering high-level results, such as successful training. Because of this, many of these techniques are infeasible for this problem. However, they do point to a possible use for statistical testing methods as applied to phenomena selection for simulations of autonomous systems for training.

2.3.3 Summary

This section discussed the area of modeling and simulation in a broader context. First, theoretical considerations from the broader area of modeling and simulation were discussed in Section 2.3.1. The main observations were that simulation models are first and foremost a collection of phenomena meant to represent a simplification of the real world. When selecting a model, we must consider which phenomena are likely to have the greatest impact on a system's behavior. This can be difficult to predict for complex systems, such as when training reinforcement learning based behaviors. As such it would be desirable

to have a method to determine the importance of individual phenomena to be modeled. However, there are no generally applicable methods from the broader area of modeling and simulation to address this.

In looking at modeling development for autonomous systems in Section 2.2, there is a broad trend towards ever increasing levels of fidelity. While in theory, increasing the fidelity of the model will always close the reality gap between simulation and the real world bit by bit, this doesn't always hold in practice. No matter how high of fidelity models are used there still seems to be a lack of transference in some sense. Section 2.3.2 looked at this from a broader modeling context and described cases where intelligent simplifications led to better results. Part of this can be understood by realizing that to get to a higher fidelity model, this generally means a greater parameterization of the model. This means more data from the real system is required to properly calibrate the behavior of the simulation. The use of default parameters or inclusion of additional assumptions to account for this missing data may actually move the model in the opposite direction, as false assumptions and misused defaults introduce new errors into the model.

As noted by Box, the goal should often be to identify the simplest model possible to achieve the current task. [10] As such, it is important to focus not only on creating a high fidelity model of the system, but to carefully select the phenomena to be considered such that the simplest model possible can be used. This is especially relevant for machine learning and reinforcement learning based systems that can be susceptible to so called "Clever Hans" behavior, where unrelated information is used as a proxy to solve the problem in a non-generalizable way. [70]

Similar observations were seen in many fields, including a close analogue to reinforcement learning: simulations for human training. In this field, it was noted that increasing fidelity could paradoxically lead to lower transference of training. In most of these cases, it was identified that the phenomena that were focused on for increasing the fidelity were not the most relevant for the task at hand. This further illustrates the point that care should

be taken to identify which phenomena are most relevant for a given goal. This is the major gap that the work described in this dissertation will attempt to address. Inspiration will be drawn from statistical model discrimination as seen in other fields.

2.4 Summary

The previous chapter discussed the rising use of autonomous systems throughout society. Much of the next generation of autonomous systems will rely on reinforcement learning for further improvements in their capabilities. This chapter was aimed at evaluating the current state of the art in this area to understand where further research is needed. First, modern approaches to reinforcement learning based policy development were outlined in Section 2.1. Major recent contributions included the application of deep neural networks to estimation of value functions for a given problem. Deep neural networks have also found applications as policy representations, allowing for powerful nonlinear policies to be found for novel systems. While difficulties remain, somewhat reliable methods for training these policies have been developed.

While attractive, these methods for reinforcement learning still largely rely on a significant exploration of the system's state-action space in order to obtain reasonable results. This exploration may lead to undesirable results, such as entering into unsafe regions of this space. This may also be prohibitively expensive as good coverage of the expected space is often required for robust solutions. To combat this, simulation-based training is often used. Approaches to this were discussed in Section 2.2. By learning in a virtual environment, the exploration can be done safely and can be completed in faster than real time. However, simulation-based training is not a panacea and brings about its own issues. These issues can largely be attributed to the reality gap between the simulated system and real system. The power of many of the modern reinforcement learning approaches becomes an issue, as policies become overfit to the simulated environments, and fail to account for changes as they move to the real world.

While some approaches have been developed to address this, there remains a large gap in the literature to be researched further. Namely, little work has gone into understanding exactly which parts of a simulation environment do allow for transference of learning, and how they can be leveraged for greatest effect. Other perspectives on modeling and simulation techniques were investigated in Section 2.3 in an attempt to fill in this gap. This resulted in two key observations. First, all models of a system are inherently simplifications. The art of modeling largely consists of defining the appropriate simplifications to make such that a model is valid for the context given while being feasible to investigate at the depth required. While methods for defining these simplifications have been developed for many system classes, this philosophical approach does not appear to have been taken for training autonomous systems. This leaves a major gap in the understanding of the relationship between modeling simplifications and the transference of learned policies for autonomous systems.

The second major observation from the broader modeling and simulation literature is that the modeling choices largely come down to phenomena representation. That is, defining phenomena as a relationship between states of an environment and the dynamic behavior of a system, the model selection problem is really a phenomena selection problem. The goal of modeling design is to determine the phenomena that most affect the resulting behavior for inclusion and omit those phenomena that have a negligible effect or present a distraction from the main goal.

All of this combined leads to a refinement of the major motivating objective behind this work identified in the previous chapter. That is, a major gap in our current understanding of modeling and simulation for autonomous systems. is the lack of a consistent method for determining the relative importance of phenomena within a given model. This leads to inefficient model development, where phenomena are continuously added to a simulation until transference can be achieved. The work discussed in this dissertation will attempt to develop a method to address the gap of measuring the importance of different phenomena

that have been proposed for a model of an autonomous system. The goal is to develop these measures in such a way that they can be used to identify possible simplified models of a system for training of autonomous behaviors. These simplified models will be evaluated based on their ability to produce transferable policies. Chapter 3 of this dissertation will discuss the first portion of this objective, measuring the relative importance of the different phenomena that can be captured by a proposed model. Chapter 4 will discuss the second half of this objective, using these measures to define simplifications for developing transferable policies. Chapter 5 will then discuss experiments to evaluate this measurement and model development strategy.

CHAPTER 3
EVALUATING PHENOMENA CRITICALITY

This work has largely been inspired by the ever increasing capabilities of autonomous systems. This has allowed for unmanned vehicles to take on greater importance for both military and civilian purposes. However, classically implemented rule-based methods of designing and developing these systems may be hitting practical limits that fall short of the possible uses for autonomous systems. To overcome this hurdle, reinforcement learning algorithms have been implemented to adapt to more complex and demanding domains. The promise of these algorithms is their generality, as they can be formulated to match with nearly any problem. [57, 71, 76, 78]

While this generality is attractive, it does come with downsides. As was discussed in the previous chapter, reinforcement learning algorithms can only learn if they adequately explore their state/action spaces. [59, 89, 111] This can lead to dangerous scenarios that are not feasible to allow in the real world. Because of this, much of the initial stages of training a reinforcement learning algorithm are conducted in a virtually simulated environment.

Virtual environments help to avoid the dangerous and potentially costly need for exploring the state/action space with a physical system, but simulation adds its own challenge of transferring learned behaviors to the real world. As many practicing roboticists are intimately aware, what works in simulation often does not work in the real world. This was true for classically developed rule-based systems and seems to be even more true for modern neural network based policies. A review of the current state of the art for transferring policies learned in simulation and the broader modeling and simulation community identified the role of modeled phenomena selection for this problem. This led to the motivating objective for the work discussed in this dissertation: to develop a method for measuring the importance of different phenomena that have been proposed for a model of an autonomous

system. The goal is to develop these measures in such a way that they can be used to identify possible simplified models of a system for training of autonomous behaviors.

This chapter will develop a research framework to help address the first half of this object: measuring the relative importance of different phenomena that could be captured by a model. First, gaps in the literature identified in the previous chapter will be matched with explicit research questions and associated hypotheses in Section 3.1. Then, an overview of the main method to assess phenomena criticality and its applications to developing simpler models will be discussed in Section 3.2. Finally, experiments to evaluate the proposed method in the defined research framework will be discussed in Section 3.3. The second half of this objective, using these measures to develop simplified models of a system, will be discussed in the next chapter and provide additional context for these measurements.

3.1 Research Framing

Chapter 1 developed the motivation behind the increasing use of autonomous systems in general, and the increasing use of reinforcement learning as a means of developing policies for these systems. The need for simulation environments was identified, and some common issues were discussed. This motivated the search for methods to improve simulation models for training reinforcement learning based policies. Chapter 2 reviewed literature relevant to reinforcement learning, sim-to-real training, and modeling and simulation in a broader sense to begin gathering further insights into these challenges.

In reviewing the relevant literature, modern reinforcement learning methods were discussed to understand how policy optimization is thought to occur. In many of these modern algorithms, exploration of the state space is paramount for finding useful policies. While previous works have shown impressive results for virtual domains, this need for exploration has ensured that a broad gap in the reinforcement learning literature with respect to systems that act in the real world remains. For these systems, the required exploration drives a need for simulation-based training for initial policy development, as broad exploration may lead

to unnecessarily risky areas of the state space if implemented on the target system. Additionally, this exploration is very data intensive, and may be prohibitively expensive to obtain in the real world. Use of virtual training models helps to alleviate these two issues.

However, this exploration is also likely to push policy evaluations towards areas of the state space where a simulation model is not accurate. The literature discussed in Section 2.2 discussed many cases where simulation based training failed to transfer to the real world. This holds true even for highly accurate models, as reinforcement learning may tend towards policies that exploit beneficial inaccuracies. Because of this, we must further our understanding of the modeling process and its affects on simulation-based training.

Literature on model development stressed the limitations of models, such that it is commonly acknowledged that "all models are wrong." One common outlook from the broader modeling community is that a model is a collection of phenomena that are thought to be critical to the input/output relation of a system. [100, 137] The larger the number of these phenomena that are captured within a model, the broader its representative space and the more accurate it can be within this space. However, our ability to represent and execute models is finite so the goal of modeling becomes identifying a limited set of phenomena that are critical to this input/output relation such that a model can be developed and executed in a reasonable amount of time. [97, 98]

As discussed in Chapter 2, much of the literature has identified the importance of phenomena representation in determining the fidelity of a model. [97, 100, 110] Similarly, there has been some work on ad-hoc approaches to rough guidelines of phenomena to capture for specific disciplinary fields. [17, 95] However, these result in limited understandings of phenomena importance, and there has been little work in the robotics community on identifying which phenomena are important to capture for transference to occur. For the remainder of this work, this importance will be defined as *phenomena criticality*. This gap leads to the following primary research question:

Research Question 1: *How can the criticality of potential phenomena to be included in a simulation model be compared such that simpler models can produce transferable policies?*

In thinking on this question, it is important to keep the broader context of a simulation model in mind. That is, the goal of a modeling an autonomous system in this case is to provide a simplified model that is faster and cheaper to evaluate than testing on the real system. Therefore, the competing objectives of simplification and fidelity must be properly balanced. By providing a useful comparator between alternative phenomena for inclusion in the model, more intelligent decisions can be made. This would allow for the development of simpler models that can still yield similar levels of transference when compared to more complex models.

In many other fields where a similar tradeoff is present, sampling based methods are used. This is true for surrogate model development and even within many RL algorithms. Consider policy gradient methods, like [106] and [71]. The true gradient is an expectation over the entire state space, with no known method to analytically derive this for complex policies. However, both works show that relatively small samples of this space can produce useful estimates of the gradient. Similarly, some multifidelity optimization techniques often employ a sampling based approach to blend results when multiple models can represent a single system interaction. [19, 85] Considering these applications to a tangential area of modeling use and within reinforcement learning itself, it is likely a sampling based approach will be adequate here as well. This leads to the following hypothesis, which will be further refined with follow on research questions throughout this chapter:

Hypothesis 1: *If a sampling-based approach is implemented to evaluate the possible simplification space, then reasonable simplified models that balance complexity with transference to the true system can be identified.*

With this hypothesis in mind, two sub-questions become immediately apparent. First, how should these samples from the simplification space be defined? The choice of sam-

pling strategy will likely have a big impact on the resulting measures of importance. This question will be discussed below in Section 3.1.1. Second, how should the simplifications that are sampled actually be evaluated? This choice includes the metrics to consider when conducting evaluations, as well as defining the actual scenarios to be simulated for a given training operation. This question will be discussed below in Section 3.1.2. Further details on the implementation and practical considerations for this method will be developed in Section 3.2.

3.1.1 Simplification Sampling

The above section provided an overview of some of the difficulties faced when trying to develop simulation models for training reinforcement learning based policies. Among the first hurdles to be addressed is identifying which phenomena to include in a given model. To do this, we first need to understand how each phenomena impacts transference, as their influence is not uniformly distributed. Due to a lack of consistent methods for this evaluation in the literature, this led to Research Question 1.0 above: how should phenomena criticality be measured? To answer this question, Hypothesis 1.0 proposed a sampling based approach.

While this is a reasonable proposal, and draws on the use of sampling based methods in many different fields, this is a bit open ended. Specifically, how should these possible simplifications be sampled? This leads to the following sub research question:

Research Question 1.1: *How should the possible simplifications of a given referent model be sampled for evaluation of phenomena criticality?*

In answering this question, it is important to consider the set of possible simplified models. This set can be represented by a multidimensional discrete space. That is, each phenomena within the referent model can be considered a dimension of the so-called simplification space. There are then two discrete locations that any simplification can take in a

given dimension: either 0, omitting the phenomena; or 1, including the phenomena. So, the null model is at the origin of the space, and the full referent model is at location [1, ..., 1]. All other simplified models lie somewhere between these two points in the simplification space.

To understand possible methods and the qualities that should be considered when sampling this space, we can look at what has been used in similar avenues of research. Namely, there is an abundance of literature on designing experiments for many types of systems. There is an important distinction that may cause issue here though: this is a somewhat meta-analysis. That is, we are not trying to evaluate what policies yield good results. Instead, we are trying to identify the models that can be used in this evaluation.

Given this slight caveat, a common approach to designing experiments for computational models is a space filling design. There are many iterations of these designs, from sphere packing to Latin-Hypercube designs. Pronzanto and Müller provide a useful overview of these designs. [92] At first blush, these are attractive methods: they attempt to uniformly sample a given space to ensure there is adequate coverage wherever the eventual optimal design appears. However, there are two primary issues.

First, these methods are largely meant to be applied to continuous spaces for parameters of a model. While this isn't an egregious limitation, it is inconvenient. Many of the space metrics commonly used would lose meaning if directly applied to the discrete space considered for simplified models. Other distance metrics that are more suited to discrete spaces, such as the Manhattan distance, could be used instead. However, even using a reasonable distance metric may miss the point: these inherently assume each dimension has an equal scale. This is another way of saying each phenomena has equal importance with respect to the fidelity of the model, which is counter to the point of this work and practical experience.

The second issue is more practical. Space filling methods tend to scale poorly with dimensions. So, for complex referent systems with many phenomena, space filling meth-

ods will find either poor solutions or become intractable. So, not only are these methods questionably applicable to this space, they may be difficult to apply in practice. Simpler methods of experimental design, such as orthogonal arrays may be an attractive alternative, but face similar issues when considering scaling and may be inapplicable due to the coupled nature of modeled phenomena.

While not directly applicable, the attractive features of past experimental designs are still relevant to this problem. As has been pointed out in much of the research on designing experiments, having a distribution of samples that roughly matches the distribution of the possible space can yield positive results. [51, 92] Similarly, dependency effects should be maintained if known and possible. [51] Taking these two features into account the following hypothesis is proposed in answer to Research Question 1.1:

Hypothesis 1.1: *If the simplified systems are sampled according to a distribution matching that of the possible simplification space, the resulting phenomena criticality measures can be used to develop lower complexity models with similar levels of transference.*

Specifics on how this sampling will be accomplished will be given below, in Section 3.2. This will be given in the broader context of a methodology for determining phenomena criticality.

3.1.2 Simplification Evaluation

As was discussed above, this work seeks to improve the development of simulation models for training behavioral algorithms for autonomous systems with reinforcement learning. One way it tries to address this problem is by asking how phenomena criticality should be measured, and a sampling based approach was proposed for this measurement. In Section 3.1.1 above, the problem of sampling the space of possible simplifications was discussed. However, there lacks a consistent method of evaluating transference throughout the sim-to-real literature. As such, there is still the open question of how each of these

sampled simplifications should be evaluated. This is captured in the following research question:

Research Question 1.2: *Given a referent and associated simplified model of a system, how should its ability to produce transferable policies be evaluated?*

Some work that is tangential to this question done by Hunter et al was discussed in Section 2.3, where the authors investigated so called "functional fidelity" of a model. [49] While this work was focused on functional models of a system, some of the conclusions may be applicable to executable models used in reinforcement learning training as well, with a bit of rework. Namely, the possible functionality space within a functional model can be considered an analogue of the possible behavioral space of a simulation model. This suggests that it would be possible to compare a model based on the behaviors that are presented during simulation with those that would be present during real implementation of the policy.

There is an important distinction to make here, though. That is, while Hunter et al were trying to build a full view of the functional spaces for comparison, only those behaviors that are likely to be seen during training are relevant for the comparison required in this work. Even simpler, direct comparison of all behaviors seen during training may be overkill or infeasible, as this is part of the motivation for using virtual environments for training in the first place. Because of this, comparisons of only the final behaviors present under the trained policy are relevant. This agrees with much of the sim-to-real literature on how only the final trained policy is evaluated for transference from simulation to the real world.

While this answers which behaviors should be used in this evaluation, there is still the question of what metrics to use in measuring this transference. To answer this question, there first needs to be a workable definition of transference with respect to policies. From the literature, there are two major metrics that are commonly used. The first will be called *Binary Transference*, and the second will be called *Performance Transference*. These two

metrics were taken from much of the literature discussed in Section 2.2 and are discussed individually below.

Binary Transference considers a qualitative evaluation of the policy once transferred to the true system. As its name implies, this is often done in the context of a success or a failure, leading to a binary evaluation. While this is useful for single attempts of transference of a one-off policy, it's less useful for evaluating a given simplification model. This is because the development and training of a policy is a random process. The exploration noise required to consistently produce reasonable policies makes the parameterization of a specific policy that results from a training scenario less reliable.

As such, it is more meaningful to consider the *probability* of Binary Transference occurring given a policy was trained on a specific model. With this in mind, the first metric of interest can be defined as the proportion of policies that transfer from a given simplified model to the referent system. That is, for a repeated number of training cases run on the same model, the Binary Transference is simply:

$$T_{Binary} = \frac{\#\, Succesful\, Transfers}{\#\, Total\, Transfers} \tag{3.1}$$

While simple, this gives a powerful measure of the quality of a given model. For cases where there is no known policy and the goal of a simulation study is to simply find a feasible policy, this is a sufficient metric. The goal would simply be to maximize the evaluated binary transference of the model. However, if the simulation study is meant to provide some sense of optimality, this metric is insufficient. That is, it does not consider the quality of a policy beyond a binary success. So, much of the sim-to-real literature uses a second metric, which will be called *Performance Transference* here.

Performance Transference is a supplemental metric once binary transference has been confirmed. This metric is simply the percentage difference between the expected value of some performance metric of the policy when implemented on both the simplified model and referent systems. For most commonly used systems, this performance metric is a function

of the trajectories generated in the given environment by that policy. That is:

$$T_{Performance} = \frac{|\mathbb{E}_{\tau\sim\zeta_R,\pi_S}[f(\tau)] - \mathbb{E}_{\tau\sim\zeta_S,\pi_S}[f(\tau)]|}{|\mathbb{E}_{\tau\sim\zeta_R,\pi_S}[f(\tau)]|} \tag{3.2}$$

For Equation (3.2), $f(.)$ is some performance measure over a trajectory, τ, through the state/action space of the environment, ζ, caused by following a policy, π. The subscripts of both ζ and π represent the system they refer to, either S for the simplified model of the system or R for the referent system. That is, ζ_R refers to the referent environment and π_S refers to a policy trained in the simulated environment. For practical purposes, these expectations are evaluated by rolling out trajectories from a sampling of initial states taken from an appropriate distribution. This metric gives a reasonable evaluation of a model's predictive abilities. That is, a model that has a lower performance transference will be better at predicting the eventual performance of a policy when it is applied to the real system.

While both of these metrics are important in evaluating the success of a given transference attempt, it is expected that Binary Transference will yield more meaningful results when attempting to simplify a referent model. First, Binary Transference is more broadly applicable. That is, most measurements of Performance Transference assume a some level of successful transference as a starting point. For example, consider the transfer of a controller for some dynamical system. In this case, stability could be considered the success criteria, while energy use is a performance metric. The performance of the controller is irrelevant if the controller is not stabilizing in the first place. As such, Performance Transference will can only be evaluated for systems where Binary Transference has already occurred.

Second, for many cases of model development, the initial selection of phenomena to include occurs at a relatively early stage in the development process. [96] So, it is likely that calibration can be conducted later to improve the predictive accuracy of the model, and therefore improve Performance Transference at the same time. Because of this, it

is expected that the use of Binary Transference as an evaluation metric will yield better decisions on which phenomena to include. This gives the following hypothesis with respect to Research Question 1.2, restated below for convenience:

Research Question 1.2: *Given a referent and associated simplified model of a system, how should its ability to produce transferable policies be evaluated?*

Hypothesis 1.2: *If the simplified models are evaluated with respect to their Binary Transference rates, then the resulting comparisons of phenomena criticality will allow for phenomena to be rank ordered in a way that allows simpler models to produce greater levels of transference.*

The specifics of actually implementing and evaluating these metrics are given below in Section 3.2. This is given in the context of a broader methodology to evaluate the importance of each individual phenomena.

3.2 Phenomena Criticality Evaluation Methodology

As was discussed in the previous section, one of the major gaps that exists in the literature is a lack of methods for comparing the relative importance of different phenomena that can be included in a simulation model. This gap exists both in the robotics focused sim-to-real literature and in the broader modeling literature. In order to address this gap, it has been proposed that a sampling-based approach could be used to extract the importance of individual phenomena from the sampled models.

There are two major assumptions that must be considered as a starting point before detailing this sampling-based method further. First, it assumes that an existing list of potential phenomena has been produced based on some referent model of the system. While the production of this list is a critical step in the modeling process, it is considered out of scope for this work. Current best practices in the context of the target system should be

followed when this list is developed.

The second critical assumption is the modularity of the modeling and simulation environment used for this study. Simply put, this method requires significant experimentation that directly alters the model of the system used in policy synthesis. If the models within the simulation are not implemented in a modular fashion, this becomes a very expensive step. The costs in developing individual simplified models from scratch may outweigh any of the benefits that could be gained from following the rest of this method.

With these two assumptions in mind, the starting point for this method can be defined. First, a target system and an associated referent model have been defined. This includes lists of phenomena that are expected to affect policy performance. Second, a modular implementation of this referent model must be developed and implemented in an appropriate simulation environment. This implementation must be developed such that the different phenomena included can be omitted as required. This also implies the ability to access and test synthesized policies on the referent model. Finally, a reinforcement learning framework needs to have been implemented such that a reasonable policy for a given model can be synthesized within the simulation. Similar to the modeling capabilities, the greater modularity within this reinforcement learning framework the better, as it will be applied to many different subsequent models.

Given this starting point, the sampling-based method developed for this thesis follows a four step process outlined in Figure 3.1. To help in understanding this method we can consider an abstract system that has four possible phenomena of interest that can be captured in a model independently. This example is intentionally designed to be abstract, as this method was designed to be applicable to a broad range of systems. These phenomena are meant to represent independent influences on the system's behavior, such as gravity, drag, etc. A system with four independent phenomena has sixteen possible modeling representations, as each can be included or omitted irrespective of the others. For this and other similar systems we can represent each of these models as a binary string such that

each character in the string represents either the inclusion or omission of a phenomena. So, the string '1010' would represent a model that includes the first phenomena and the third phenomena, with the second and fourth phenomena omitted. For further clarity, the string '1111' would represent the full referent while the string '0000' would represent the null model. At this stage, the ordering of the phenomena is entirely arbitrary and should have no effect on the eventual outcome of this analysis.

Given these representations of simplifications, the models can be sampled from the possible simplified model space according to a given distribution. Continuing the example of a system with four candidate phenomena, this would entail selecting a number of the 16 different possible combinations to evaluate. Say we take 4 models in our sample, defined as the models represented by the strings 0101, 0011, 0110, and 1001. At this point, each sampled model can be implemented, used in training a policy, and the transference of these policies can be evaluated. This training can be done in whatever reinforcement learning framework is appropriate for the system considered. As this work is most concerned with continuous state and action spaces, the DDPG framework will be used. [71] For further details on this framework and adjustments made for asynchronous training, please see Chapter C.

To extract the influence of individual phenomena, the next step is to classify these models into non-exclusive sets defined by the inclusion of a set's characteristic phenomena. This is defined below:

$$\mu_i = \{m \in M : p_i \in m\} \tag{3.3}$$

That is, each set, μ_i is defined as the collection of models, m, within the set of all sampled models, M, that contains the i^{th} phenomena, p_i. Continuing the example above, the first set of models, μ_1, would only contain the fourth model, 1001. The second set, μ_2, would contain the models 0101 and 0110. The third set would contain the models 0011 and 0110. The fourth set would contain the models 0101 and 1001.

With trained policies and predictions of performance of these policies for each individual simplification, we can then test transference to the referent system. As defined in Section 3.1.2, both Binary Transference and Performance Transference will be measured. The average transference to the referent within a group, controlled for fidelity levels of the individual models, will be used to define the criticality of the characteristic phenomena. The phenomena can then be ranked according to this criticality measure.

In order to test Hypothesis 1, then, we must use these measures of criticality to define a series of models of increasing complexity. This can be done by simply including the phenomena in decreasing order of criticality. That is, first build a model that includes the most critical phenomena. Then, build a second model that includes the two most critical phenomena, and so on. The models constructed in such a way should show greater transference with fewer phenomena included. Further details of questions and hypotheses with regards to this model construction can be found in the next chapter, Developing Simpler Models.

Given this overview of the four step method, the remainder of this section will detail the individual steps of the proposed method. This will include discussion of decisions within each step, and their expected impacts on the overall results.

3.2.1 Sample Simplified Models From Referent

The first step is the philosophically simplest step: sample possible simplifications of a referent model. Given the list of phenomena to investigate for the referent system, we can define a simplification space. Each phenomena within the referent model can be seen as a dimension. A simplified model of the referent can then be defined by setting each dimension to either 0, where the phenomena is omitted, or 1, where the phenomena is included. This space forms a discrete set that contains all possible simplified models that can be constructed from the individual phenomena within the referent list. The goal is then to find a model within this set which balances model simplicity and transference to the true

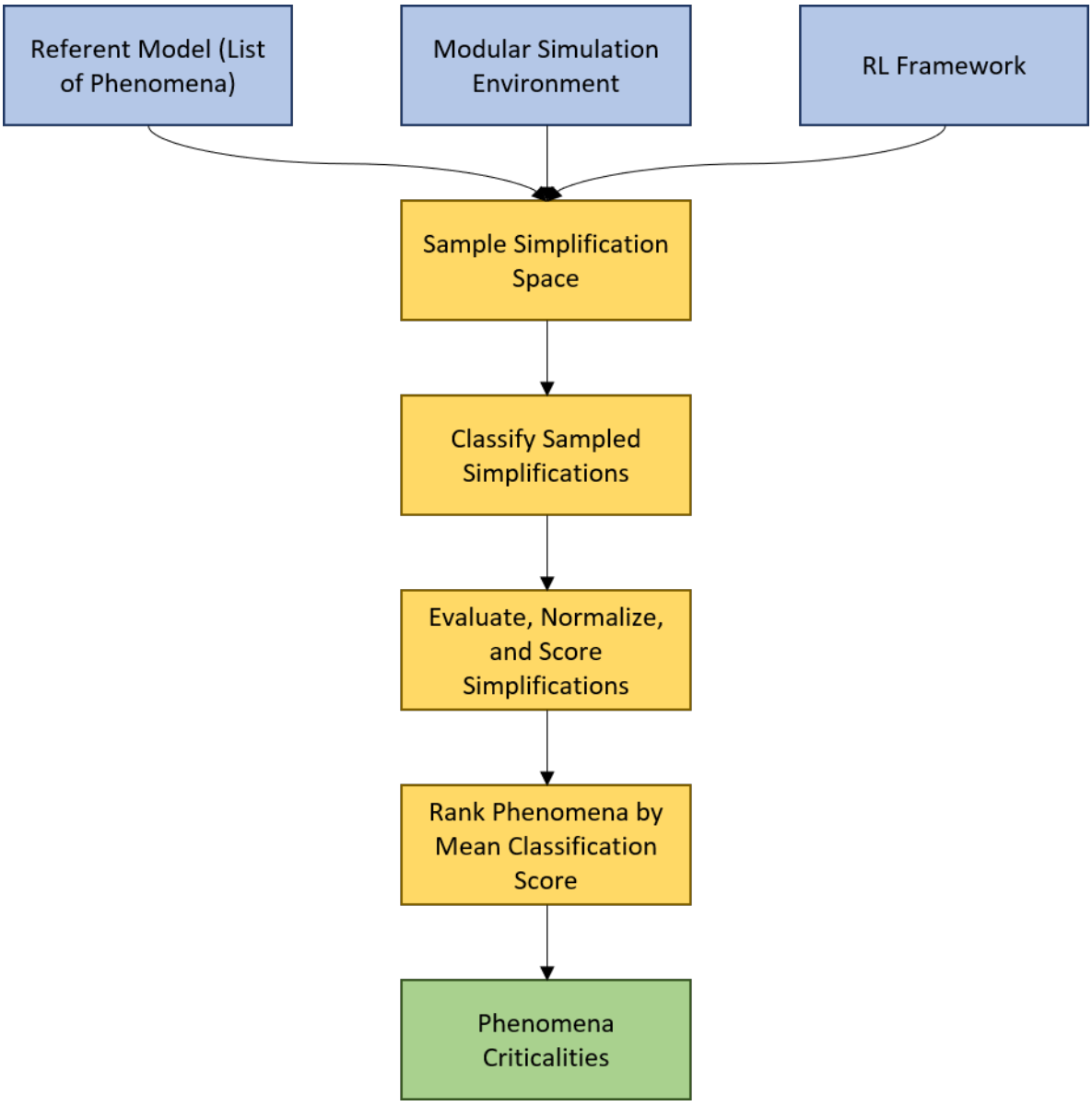


Figure 3.1: An overview of the developed method. First, the simplification space of for the identified phenomena is sampled to define a design of experiments. The simplified models associated with this design are implemented, and a policy is synthesized for and evaluated for each. These policies are then transferred to a referent model (which can be the truth system) and evaluated. The resulting data is categorized into nonexclusive sets for each phenomena. If a simplified model included a phenomenon, its data is included in this set. The transference statistics for each set are compared, and a relative ordering of the phenomena is produced.

system.

This set scales exponentially with the number of phenomena considered. For relatively simple referents, this is a small set and it may be feasible to search the space in a full factorial method to find a suitable model. However, this set can quickly grow such that it would be impossible to consider all possible models. Consider a model with 32 phenomena of interest. This set would contain over 4 billion possible simplifications, much too great for a full factorial search to be completed in a reasonable amount of time. As such, a method for sampling this modeling space is required. This led to Research Question 1.1, which asks how these samples should be taken.

When considering different approaches to answering this question, there are a few important things to keep in mind. First, any sampling strategy must be feasible to implement given restricted computational resources. It must also be done in a way to maximize information gained about the importance of the individual phenomena to be considered. These two goals are in opposition: constrained resources pushes towards fewer samples to be taken while information gain pushes towards a full factorial search. The sampling method chosen must balance these two concerns appropriately.

As mentioned above, much of the design of experiments literature has discussed the importance of using samples that are representative of the space. This representative sampling should allow for a comparatively sparse sample to provide sufficient information to differentiate between the relative importance of the different phenomena the model may contain. This was the foundation for Hypothesis 1.1.

Taking the view of the simplification space as outlined above, it is clear that any representative sampling of this space should include each individual phenomena with a probability of 50%. To see this, consider a full-factorial search of the space. Each phenomena can be considered independently from all others. So, the number of simplifications that include a phenomenon will equal the number of simplifications that omit it. When taking a simplistic view of fidelity as the number of phenomena included, this yields a distribution

of model fidelity similar to the expected distribution of a number of Bernoulli trials with $p = 0.5$ and the number of trials equal to the number of phenomena considered. That is, there will be a peak with roughly half the phenomena considered and a relative lack of models sampled at extremely high or low fidelity.

By sampling at greatest density in the moderate fidelity range, a second possible benefit is likely to be achieved. Low fidelity models are likely to transfer at very low rates, making it difficult to distinguish between the effects of phenomena. That is, the total fidelity of the model will dominate the transference rates, limiting the effects of the inclusion of individual phenomena. Conversely, it is reasonable to expect that high fidelity models will lead to high transference rates. Again, this may be driven more by the total fidelity of the models, not the individual phenomena included leading to less distinguishing information available. Moderate fidelity models will likely lead to moderate transference rates. This means that there will likely be greater variance in the transference rates if the phenomena contribute disproportionately to transference. So, fewer samples are expected to yield greater information on the importance of the individual phenomena.

We can achieve this representative sampling using a fairly simple computational strategy. If we uniformly sample the integers from the set $\left(0, 2^N - 1\right)$, where N is the number of phenomena making up the referent, and take their binary representations, we achieve this distribution.To see this, consider the most significant digit of the binary representation. It will be 1 if the sampled integer is greater than 2^{N-1}. It will be 0 if the sampled integer is less than or equal to 2^{N-1}. On the interval $\left(0, 2^N - 1\right)$, each of these classes has exactly $2^{N-1} - 1$ members, and so has probability of occurring of 50%. Looking at the second most significant digit, we now look at the sampled integer modulo 2^{N-1}. Similar to before, if this is greater than 2^{N-2}, this digit will be 1. If it is less than or equal, it will be 0. Again, each of these sets is half of the possible space, and so yields a 50% probability of the second phenomena being selected in a model. This can be repeated until it is shown that all individual phenomena have a 50% chance of being selected.

In this way, we can now sample simplifications from the full simplification space while maintaining a representative distribution. Continuing the example of a system with four phenomena from above, we take a sample of 4 possible simplifications, represented by the strings 0101, 0011, 0110, and 1001. For each model, a policy can be trained and transference evaluated. If we aggregate the results of the policies trained on each of these models, we might expect something such as that seen in Figure 3.2, showing notional results for a model with four possible phenomena. While useful, this aggregate curve cannot be used to evaluate individual phenomena. The next section will discuss how these models will be classified in order to evaluate the effects of individual phenomena.

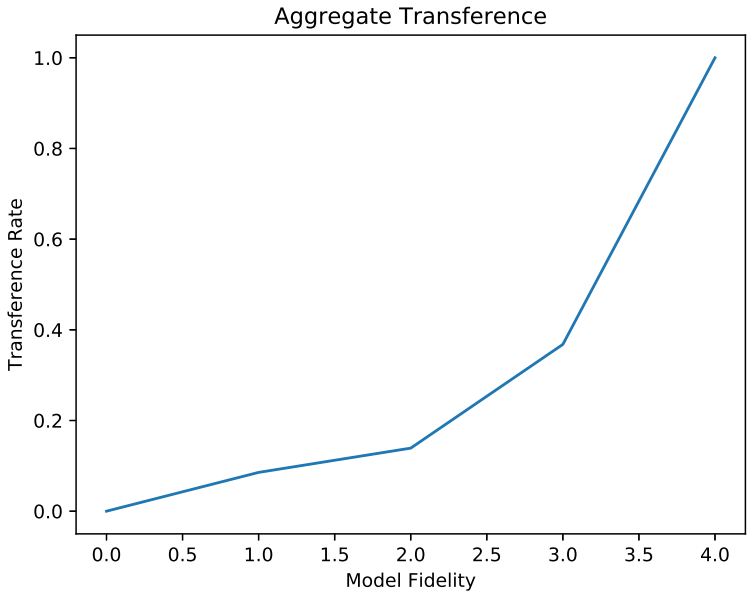


Figure 3.2: A notional transference curve for models sampled from a referent system with four possible phenomena to be represented.

3.2.2 Classify Simplifications by Characteristic Phenomena

The previous section discussed a method for sampling these strings in a representative manner such that each each individual phenomenon is included at a uniform rate, and the fidelity of the sampled models is weighted towards moderate fidelity levels. Following

from the inspiration from ANOVA based methods, the sampled models will be classified into sets based on the phenomena they use. These sets will be characterized by a single phenomena, and all contained models will include this phenomena in conjunction with others. This will allow for the effects of a given phenomena to be evaluated by looking at the mean performance of the models within the set. These sets are defined according to:

$$\mu_i = \{m \in M : p_i \in m\} \tag{3.4}$$

That is, each set, μ_i is defined as the collection of models, m, within the set of all sampled models, M, that contains the i^{th} phenomena, p_i.

We can make this more explicit by continuing the example above with four possible phenomena. This gives sixteen possible simplifications (including the null and full referent models) that can each be represented by a binary string of length four. Each character in the characteristic string representing a model signifies whether that phenomena is included in the model or not. As before, we consider a sample of four of these models, represented by the strings 0101, 0011, 0110, and 1001. The set of models representing the first phenomenon would only contain the model defined by the string 1001, as this is the only sampled simplification that includes the first phenomenon. The set of models representing the second phenomenon would have the models defined by 0101 and 0110. The set of models representing the third phenomenon would have the models defined by 0011 and 0110. The set of models representing the fourth and final phenomenon would have the models defined by 0101 and 1001.

Similar to before, we can then look at the aggregate transference for each of these sets. A notional example of this is seen in Figure 3.3 for a model with four possible phenomena. Now, we can see a distinction between the phenomena that contribute significantly to transference and those that do not.

The overarching assumption behind classifying models this way, that the impacts of an individual phenomena can be assessed by comparing aggregate evaluations of a character-

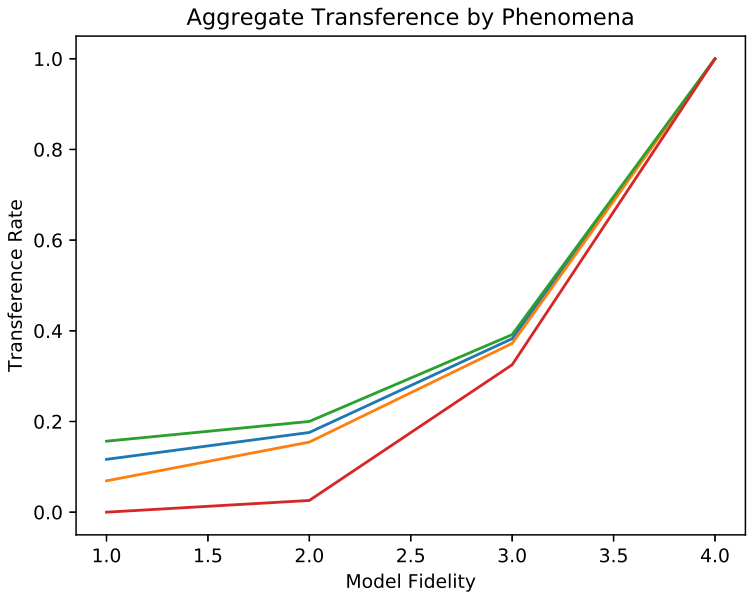


Figure 3.3: Notional transference curves for models sampled from a referent system with four possible phenomena to be represented. Each curve represents a set of models characterized by a different phenomenon.

istic set, is common in many methods of experimental analysis. The most direct inspiration for this would be ANOVA tests, and the similar Kruskal-Wallis test for non-parametric models, as applied to effect analysis for experiments involving applications of different treatments.

While it would be tempting to simply apply these tests, ANOVA and other similar statistical tests cannot be applied directly in this case, as each of the main assumptions leading to these analyses are violated. The largest violation is in the construction of the sets themselves. For ANOVA and similar tests, they require the different classifications to remain independent. That is, the sets should be mutually exclusive. As shown in the example above, that is not the case here.

While it would be possible to construct a set of rules to assign the models such that each set is mutually exclusive, it is expected that this would have two significantly negative effects. First, this would greatly increase the sampling density required to have similar classification sizes. This is because the degree of overlap is significant as nearly all models

will belong to at least two classifications. The second effect would be a weakening of the test, as it would introduce significant confounding variables. It would be possible to control for the remaining phenomena used in a model, but this would yield poorer results at greater cost than simply constructing the classifications in the proposed non-exclusive manner.

The other assumptions that have been violated for these tests are:

- Lack of significant outliers: It is reasonable to expect some combinations of phenomena to to yield significantly better or worse predictions of system behavior than others. These outliers will have an outsize effect on the resulting variances leading to less discriminating power.

- Normality: In cases of non-additive phenomena, there is no reasonable expectation for normality in the distribution of transference. The possible coupling between phenomena is likely to lead to noticeable groupings of models.

- Homogeneity of variances: Assuming a non-uniform distribution of importance among the phenomena of interest, the variances should be directly correlated with importance. To see this, consider the group that is characterized by the least important phenomena. The most important phenomena will be included in roughly half of these models and omitted from the other half. This leads to higher variance than the group characterized by the most important phenomena.

However, even given these limitations, this general framework should still give useful information. The spirit of the means discrimination tests remains intact. Given the high confidence in the underlying assumption that different phenomena contribute to transference in a non-uniform fashion, the direct comparison of the non-exclusive set means should be sufficient for discrimination. The next section will now discuss how each of these models will be evaluated such that these classifications can be evaluated on the aggregate.

3.2.3 Score Transference Metrics For Simplifications

The previous section discussed the classification of models sampled according to a representative distribution defined earlier. These non-exclusive classifications are characterized by the inclusion of a characteristic phenomenon. This section will discuss the evaluation used for each sampled model such that the aggregate performance of a classification can be used as a measure of the criticality of its characteristic phenomenon.

As previously discussed, there are two primary metrics used in the literature when discussing transference. The first was *Binary Transference*, defined in Equation (3.1). This is an estimate of the probability that a policy trained on a simplified model of a system will successfully transfer to the referent model of the system. The second primary measure from the literature was *Performance Transference*, defined in Equation (3.2). This can be seen as a measure of the simulation model's ability to predict the behavior of a policy once it has been applied to the real system.

Each of these measures has varied benefits and detriments. Performance Transference provides a higher resolution quantification of the performance of a simulation, and is more similar to the common approaches of verification and validation common through much of the modeling and simulation literature. However, it is less broadly applicable, as its measure assumes an inherent level of successful transference in the first place.

In contrast, Binary Transference is more broadly applicable as it can be applied to any policy without arbitrarily assigning values to unsuccessful attempts for transference. Similarly, its qualitative nature is more in line with the early stage of model development where decisions of phenomena inclusion and omission are most relevant. That is, it is reasonable to expect Performance Transference can be increased by later calibration to truth data, while this will be more difficult with Binary Transference. As such, Binary Transference is hypothesized to yield better results within the above methodology. Given this hypothesis, each sampled model will be evaluated by first training a policy through simulation of the simplified model until it is considered successful. This policy will then

be implemented on the referent model and tested for success. As stated above, this will yield a binary measure for each individual model.

In order to get a probability of transference then, each model of the same fidelity level (taken to mean the total number of phenomena included in the simplified model relative to the total number of phenomena listed for the referent) can be compared. This will give an estimate of the probability of transference for that fidelity level. The individual models will then be judged relative to this probability of transference. This is shown in Figure 3.3.

However, we cannot simply compare these results directly, as it's possible that some phenomena will be over-represented in high-fidelity models and some phenomena will be over-represented in low fidelity models. Similarly, while Figure 3.3 shows a notional example of the expected results, it is like the separation between phenomena classifications won't be as clear cut. To account for this, we must normalize these transference measures by fidelity level. Various methods exist for this, but we will use the Standard Scaling, or Z-score method. This will transform the distribution of models at each fidelity level such that it will have zero mean and a standard deviation of one. This can be seen applied to notional transference curves for the ten phenomena system in Figure 3.4.

Continuing the example above of a system with four phenomena, with the four of sixteen sampled models defined by the strings 0101, 0011, 0110, and 1001. Say the first and third models, those defined by the strings 0101 and 0110, both successfully transfer while the other two do not. Since each sampled model contains two phenomena, the estimated probability of transference for two-phenomena models is 0.5, with a standard deviation of 0.5. So, the first and third models will receive a normalized score of +1, while the second and fourth models will receive a normalized score of -1. This straightforward analysis is expected to give the information necessary to now evaluate the classifications as a whole, as discussed in the next section.

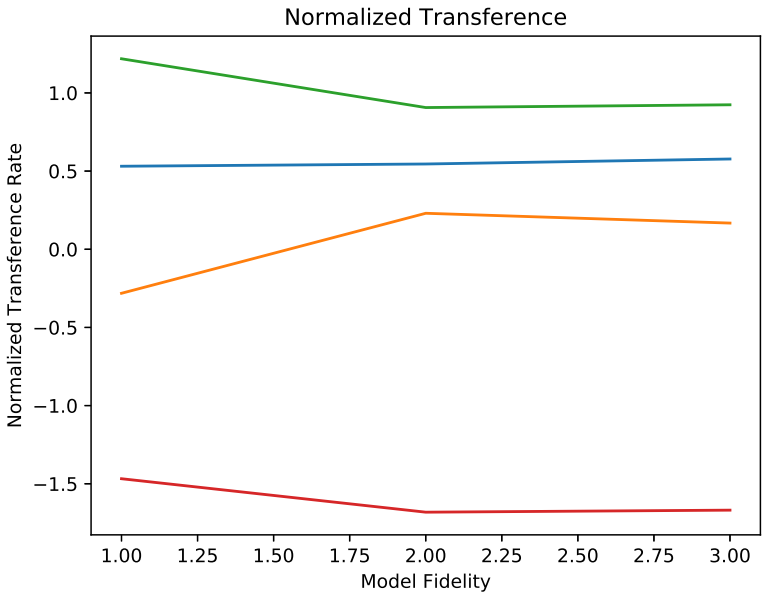


Figure 3.4: Notional transference curves that have been normalized by fidelity level for models sampled from a referent system with four possible phenomena to be represented. Each curve represents a set of models characterized by a different phenomenon.

3.2.4 Rank Phenomena By Simplification Scores

The previous section discussed the evaluation of single models that were sampled according to the representative distribution discussed in Section 3.2.1. This resulted in a z-score for each individual model such that it was normalized by comparisons to models with similar fidelity levels. This section will now discuss a straightforward aggregation of these normalized scores for each classification as discussed in Section 3.2.2.

While many aggregation techniques could be used, a simple mean comparison is expected to be sufficient here. More complex techniques are often helpful when it is unsure if the characterizing features of a classification actually lead to varying distributions for measures of performance. However, as was laid out above and in the previous chapter, it is already well accepted that different phenomena contribute different amounts to the representation of behavior for a given system. So, each classification will be scored simply by taking the mean z-score of each of the models contained within it. These classification

scores are then attached to the associated characteristic phenomenon as the final measure of criticality.

For clarity, we can finish the example that has been carried through the previous sections. The first phenomena was used to characterize the set containing only the fourth sampled model. This model received a normalized score of -1, so the first phenomena is given a measure of criticality of -1. The second phenomena characterized a set containing the second and third sampled models, each receiving scores of +1, yielding a criticality of +1. Both the third and fourth sets yielded mixed scores, so both the third and fourth phenomena are scored with a criticality of 0. So, the second phenomena was the most critical to capture, the third and fourth phenomena were equally important, and the first phenomena was the least important to capture.

It is expected that this method will generalize well to many different classes of systems. The following section will now discuss the design of a series of experiments to evaluate this method in the context of the research questions and hypotheses that have been proposed above.

3.2.5 Summary

This section laid out the proposed methodology for measuring phenomena criticality. This method follows a four step process, shown in Figure 3.1. To begin, this method requires a list of possible phenomena to define a referent model of the system, a modular simulation environment, and a reinforcement learning framework for policy synthesis.

Given this starting point, the first step of the method is to sample simplified versions of the referent system, detailed in Section 3.2.1. This is done by sampling integers from the open interval $(0, 2^n)$ according to some distribution. The binary representation of these integers then define a set of simplified models to implement and train policies on. Each of these sampled models can be implemented and policies synthesized.

The next step of this method is to classify these simplifications into sets that represent

the influence of individual phenomena, detailed in Section 3.2.2. This is done by creating non-exclusive sets characterized by the inclusion of each phenomena individually. This will produce a set of all the sampled models that include the first phenomena, a set of all the sampled models that include the second phenomena, and so on. Clearly, there will be overlap between these sets, as the majority of the sampled models will contain multiple phenomena. However, it is expected that aggregating results according to these sets will still allow for the influence of the individual phenomena to be extracted.

Following this classification into characteristic sets, we can then evaluate policies produced by each of the sampled models for transference to the referent system, detailed in Section 3.2.3. This will yield a set of transference curves, one for each phenomena, similar to the notional example shown in Figure 3.3. However, to get an accurate measure of criticality for each phenomena, we must normalize these transference scores by fidelity level. This is done using z-scores as a standard scaling approach, shown for the notional example in Figure 3.4

At this point we have transference scores for each sampled model, all normalized by fidelity of the model. The final step in measuring criticality, detailed in Section 3.2.4 is to aggregate these scores according to the classifications defined in the second step. This can be done by simply taking the mean normalized transference score across each set. These aggregated scores for then can be used as the criticality measure for the phenomenon that characterizes the set. The phenomena can then be ranked according to criticality and used in developing simplified models of the referent.

3.3 Experimental Definition

This chapter has focused on developing a methodology around measuring the importance of different phenomena that can be used to develop a simplified model of a referent system for training reinforcement learning based policies in simulation. First, a general framework for researching this problem was developed in Research Framing. Then, the methodology for

measuring these importance levels, here called phenomena criticality, was further detailed in Phenomena Criticality Evaluation Methodology. This section will now discuss experiments that will be used in evaluation of this research framework. First, a general proof of concept will be described to begin evaluating Hypothesis 1.0 in response to Research Question 1.0. This will combine the proposed framework with a simple model development strategy and compare the resulting curves for transference at different fidelity levels with those of comparative baselines.

Assuming a successful proof of concept through this experimentation, two follow on experiments will be conducted. First, the effects of different sampling strategies and the density of samples taken. This will be focused on evaluating Hypothesis 1.1 in response to Research Question 1.1. The second of these experiments will evaluate the effects of using different low level metrics when evaluating the criticality of a model's phenomena. This will be focused on evaluating Hypothesis 1.2 in response to Research Question 1.2.

The following sections will describe these experiments in further detail.

3.3.1 Experiment 1: General Proof of Concept

This chapter has focused on developing a methodology around measuring the importance of different phenomena that can be used to develop a simplified model of a referent system for training reinforcement learning based policies in simulation. This was framed in response to the question *How can the criticality of potential phenomena to be included in a simulation model be evaluated such that simpler models can produce transferable policies?* It was hypothesized that a sampling based method as laid out in Section 3.2 would yield an evaluation of phenomena criticality that can be used to improve the transference of simplified models of a referent.

In evaluating this question, the second half of this research question deserves special attention. That is, the context suggested by "*such that simpler models can produce transferable policies*" implies this method cannot be evaluated in a vacuum. The eventual output

criticalities from this method must be used to develop simulation models of increasing complexity, and the transference from these models should show improved transference relative to models of similar complexity whose phenomena were chosen through alternative methods.

That is, there is no truth data that can be used to determine whether the produced phenomena criticality measures are accurate. We can only evaluate their usefulness in the context of deciding which phenomena to include in simplified models. So, this evaluation method must be paired with a model development strategy in order to be evaluated. This strategy will be further developed in the next chapter, as well as additional baseline strategies to compare against, but suffice it to say that a simple method should work well if the criticality measures are accurate.

In this case, the simplest strategy possible would be to simply develop models of increasing complexity by including the next most critical phenomena. This can be illustrated by continuing the example from the previous section with a system with four phenomena to consider. As a refresher, the second phenomena was found to be the most critical, the fourth phenomena was found to be the second most critical, the third phenomena was found to be the third most critical, and the first phenomena was found to be the least critical. This would mean our models of increasing complexity would be defined by the strings 0100, 0101, and 0111. The full referent could also be included if desired to give one model for each possible fidelity.

Each of these models would be evaluated for transference of policies to the referent system. This will result in a curve of the transference metrics against fidelity[1] of the simplified models. These models will be evaluated for both Binary Transference, defined in Equation (3.1), and Performance Transference, defined in Equation (3.2).

Both of these metrics give reasonable measures of a simplified models ability to produce transferable policies. However, they largely measure the accuracy of predictions of

[1] Again, fidelity refers simply to the number of phenomena included in the simplified model here.

performance. These predictions do not necessarily define the capability of a given model to actually train the policy. To address this, a third metric, called *Potential Transference* is proposed. This takes the form:

$$T_{Potential} = \frac{\mathbb{E}_{\tau\sim\zeta_R,\pi_R}\left[f\left(\tau\right)\right] - \mathbb{E}_{\tau\sim\zeta_R,\pi_S}\left[f\left(\tau\right)\right]}{\left|\mathbb{E}_{\tau\sim\zeta_R,\pi_R}\left[f\left(\tau\right)\right]\right|} \tag{3.5}$$

The difference between Equation (3.5) and Performance Transference, Equation (3.2), is subtle and may not be noticed at first glance. While the form is the same, the evaluation of the expectation for the referent environment is now conditioned on a policy that has *also* been trained on this ***referent*** environment. That is, the performance of the algorithm trained in simulation is evaluated after being transferred to the true system. Instead of comparing this with the predicted performance, it is compared with the actual optimal performance that could be achieved through training on the referent system.

In most cases, given sufficient training time, an algorithm trained directly on the referent system will perform better than a policy that was trained on a simplified system. This is due to two major factors. First, advantageous phenomena that were simplified out of the model can be leveraged. Second, disadvantageous phenomena that were ignored in the simplified model can be properly accounted for. These two factors can lead to greater performance. As such, this metric provides an added dimension to give a fuller picture of the quality of a simplified model. While a useful metric, this can only be evaluated for simple referent systems. As such, this experiment, and the following experiments meant to evaluate different decisions to be made when implementing this methodology will be conducted on a fairly simple system.

To simplify comparisons between alternatives further, the area under these three fidelity-transference curves will be used as the final comparison metric. This avoids qualitative comparisons of curves, and can properly balance the desires to produce early transference with quality of the transferred policies at higher fidelities. For binary transference, the ideal measure for this metric would be 1.0. This would imply that the models produced

by the simplification development methodology produces models that transfer to the referent as soon as the most important phenomena is included. Clearly, this metric is system dependent, as some systems will likely present a greater challenge to transference than others. Even in this case, comparison between methods using this metric is meaningful as the greater this measure, the greater the transference achieved at lower complexity levels.

This measure is also useful in evaluating performance and potential transference. However, as these two measures are related to the difference between the simulated policy and policies evaluated or trained in the real world, the goal is to now minimize them. That is, the ideal simulation model would achieve both perfect prediction and full training with the fewest phenomena accounted for. This would yield an area under the curves of 0. This ideal case is unlikely to happen in the real world though, so comparisons of different methods will largely come down to which yielded a lower value.

So, to evaluate the usefulness of the phenomena criticality method, it must be implemented on a referent system. This will result in a ranking of the different phenomena captured within the referent. These rankings can then be used to develop increasingly complex simplified models. Evaluating the transference of these models will provide a fidelity-transference curve for the three major metrics identified. By taking the area under these curves, different methods for evaluating phenomena criticality and selection methods can be compared directly. Increasing the area under the binary transference curve and decreasing the areas under the performance and potential transference curves determines an improvement. If the basic model development strategy discussed in the next chapter produces favorable results compared to alternatives, Hypothesis 1.0 will be considered supported.

3.3.2 Experiment 2: Effect of Sampling Strategy

The previous section developed a proof of concept experiment to evaluate the potential usefulness of this method. One of the most important points noted in developing this ex-

periment was a recognition that these experiments must be evaluated in the full context of developing models and attempting transference of policies from these simplified models to the referent system. This point is also important to consider in evaluation of sub portions of the methodology. That is, any changes to sub portions of the methodology must also be evaluated in the full context of developing a set of increasingly complex simplified models.

The first of these sub portions to be evaluated will be the sampling strategy use in selecting simplifications to evaluate. This was stated in Research Question 1.1, and it was hypothesized that a representative sampling distribution of simplified models would be most useful within this methodology. The representative sampling strategy was described in Section 3.2.1. In essence, this sampling strategy was defined to to include each individual phenomenon with uniform probability. While this is simple and supported by some of the literature, it is not the only possible strategy.

As was noted previously, this representative strategy mimics the distribution of the full simplification space. This leads to a heavy weighting towards moderate fidelity levels. As such, the alternatives will largely be focused on generating alternative distributions in the fidelity space. The first alternative sampling strategy will enforce a uniform distribution over the fidelity of the sampled models.[2] This will be accomplished by using a hierarchical sampling strategy. First, the fidelity of the sampled model will be drawn from a uniform distribution. Then, the a sample containing an equivalent number of phenomena from the full list of phenomena in the referent will be drawn. This idea behind this method is that there is valuable information at both the extremely low and extremely high fidelity ends of the simplification space. If this is true, this method should outperform the representative sampling strategy originally hypothesized.

The second alternative method will follow a similar idea. This time, however, the idea is that the most information is concentrated only in the high fidelity end of the simplifica-

[2]There is an important distinction to note here. For some models and numbers of samples, it may be impossible to make a truly uniform sampling of the fidelity space. This is because the extreme ends of the fidelity space severely limit the absolute number of simplifications that exist. So, while it will be referred to as a uniform distribution for simplicity, it is in actuality a quasi-uniform distribution.

tion space. As such, a triangular distribution over the fidelity of the simplified models will be used, with the extreme end representing the referent system.[3] This will also be accomplished using a hierarchical sampling strategy. First, the fidelity of the simplification will be sampled from the triangular distribution. Then, the equivalent number of phenomena will be sampled from the full referent list of phenomena.

With these alternative sampling strategies defined, the evaluation will proceed largely the same as the proof of concept experimentation. Each sampling strategy will be implemented within the methodology to determine the rankings of possible phenomena. These rankings will be used to develop a series of increasingly higher fidelity simplifications. These simplifications will be used to train policies that will be evaluated on the referent system, producing a fidelity-transference curve for the binary, performance, and potential transference metrics defined above. The areas under these curves will be compared to determine which strategy is superior.

One additional piece of information will be used in this evaluation to provide additional information: the number of samples used by each strategy where the criticality measures begin to converge to their final evaluation scores. This is an important piece of information for comparison, as each strategy should converge to the same criticality rankings given enough samples. To see this, we return to the example system with four phenomena that was previously discussed. This system has 16 possible simplifications. If we take 16 non-repeating samples from the simplification space, we will always receive the same description of the full space regardless of the sampling distribution used. As such, we don't care only about which sampling strategy yields the best eventual ranking, we also care about how quickly it reaches this ranking.

[3] Similar to the previous note on the uniform distribution over fidelity of the models, this is more accurately stated as a quasi-triangular distribution.

3.3.3 Experiment 3: Effect of Evaluation Metrics

The previous section described an experiment to evaluate the sampling strategy used in selecting simplifications to evaluate. This section will now look at how the low level metrics used in evaluating transference from specific simplified models to the referent model effect the eventual measures of phenomena criticality. This is meant to test Hypothesis 1.2 in response to Research Question 1.2. That is, it was hypothesized that using Binary Transference as the low level metric would produce measures of criticality that lead to improved transference for the developed simplifications when compared with using Performance transference as the low level metric.

In developing a proof of concept experiment to evaluate the potential usefulness of this method in Experiment 1: General Proof of Concept, one of the most important points noted was a recognition that this method must be evaluated in the full context of developing models and attempting transference of policies from these simplified models to the referent system. This point is also important to consider in evaluation these low level metrics.

This holistic evaluation will follow essentially the same format as experiment described in the previous section for evaluating the effects of different sampling strategies. That is, the nominal phenomena criticality evaluation method described in Section 3.2 will be implemented. This will actually be two implementations, once with Binary Transference as the scoring metric used when evaluating individual simplified models that have been sampled, and once with Performance Transference as the scoring metric. Each of these implementations will be used to determine the ranking of the phenomena for a referent model. These rankings will be used to develop a series of increasingly complex simplifications. These simplifications will be used to train policies that will be evaluated on the referent system, producing a fidelity-transference curve for the binary, performance, and potential transference metrics defined above. The areas under these curves will be compared to determine which low-level scoring metric is superior. Specifically, if one metric fully dominates the other it will be considered strictly superior. However, if there is a case of non-dominance

between the two metrics, the results of of Binary Transference will take precedence. This follows from the same logic as defined in Section 3.2.4 when it was hypothesized that Binary Transference would be a more useful metric.

One thing to note in this is that Potential Transference, defined in Equation (3.5), will not be considered as the scoring metric. This is due to the difficulty in evaluating Potential Transference on realistic systems. As such, it would be difficult to recommend its use when the main motivating goal was the inaccessibility of the true system.

3.4 Summary

When discussing the simulation based training literature in the previous chapter, one major theme was recognized. When it came to developing simulation models, a "more is better" approach was often taken. That is, simulation models are often developed by including as many phenomena representations that may affect system behavior as feasible with little effort spent understanding which phenomena are most important to capture. Ad-hoc approaches to model development like this have struggled to produce simulation models that consistently train reinforcement learning based policies that then transfer to the real world.

This gap in the literature with respect to evaluating the importance of various phenomena to be captured in a simulation model motivated the development of Research Question 1. In response, it was hypothesized that a sampling-based approach for evaluating potential simplifications could be used to extract the criticality of individual phenomena. This was formalized in Hypothesis 1.0.

In laying out the sampling-based methodology for Hypothesis 1.0, two major decisions were discussed. The first was in reference to the distribution from which to sample the possible simplifications. This distribution could take many forms, and parameterized in many ways. This led to the Research Question 1.1, asking which distribution would be most useful for the method. Here, Hypothesis 1.1 proposed a simple approach to sampling the simplification space that results in a distribution of similar form as the space itself was

proposed.

With a proposed distribution for simplifications to be sampled from, the second major decision to be made is how the sampled simplifications should be evaluated. While potentially expensive, direct evaluation of trained policies was expected to be most useful based on the relevant literature. At the low level, it was hypothesized that looking at Binary Transference as defined in Equation (3.1) was likely to yield the most valuable information due to its broad applicability. This was in contrast to Performance Transference, defined in Equation (3.2), which was more narrowly applicable but would yield greater per-model information.

With this framing, 3 experiments were discussed. The first was a simple proof of concept to demonstrate the methodology and evaluate the feasibility of Hypothesis 1.0. In this, the methodology for phenomena criticality detailed in Section 3.2 will be paired with a basic model development strategy to see if models developed through this method will lead to less complex models that allow transferable policies to be trained. This model development strategy and other practical implications will be discussed further in the next chapter, Developing Simpler Models. A second experiment evaluating Hypothesis 1.1 was also discussed. This will largely look at the impact of different sampling distributions and the density of samples taken. Finally, a third experiment evaluating Hypothesis 1.2 was discussed. This will investigate the effects of using different low level metrics for evaluating the sampled models.

An important point was made when discussing these experiments. That is, the method for measuring phenomena criticality cannot be evaluated in a vacuum. It must be evaluated in the context of model development. Otherwise, there is no sensible way to evaluate whether the criticality measures will actually lead to simpler models, or alternatively if models of similar complexity developed through this method will lead to greater transference. To address this further, the next chapter will discuss a basic approach to simplified model development and some practical aspects that must be dealt with. This will then build

upon these developments to extend the experiments described in Section 3.3 to solidify how they will be evaluated.

CHAPTER 4
DEVELOPING SIMPLER MODELS

The previous chapter, Evaluating Phenomena Criticality, discussed the proposed method for measuring phenomena criticality within the context of a simplified model of an autonomous system used for training reinforcement learning based policies. That is, the goal of the method discussed in the previous chapter was to provide a measure of the relative importance of each phenomena that could be included within the simplified model. While having the ability to make these measurements is crucial, it is only half of the problem. The other half is actually using this metric to develop and use the simplified models to train autonomous policies. This includes defining how the simulation models should be defined and dealing with practical considerations that will be seen in realistic systems. This chapter will develop the research framework around this half of the problem.

In approaching this, it is useful to see the questions raised here as adding necessary context to the evaluation of phenomena criticality. It would be nonsense to test a phenomena criticality evaluation method in a vacuum, as there is no way of validating the resulting measures directly. The goal of producing simplified models will be discussed by returning to the motivation for simplifying models. This and the effects of using possibly incomplete information when developing the referent model will be framed in Section 4.1. Experiments meant to evaluate the choices made in generating simplified models will be laid out in Section 4.2. This framing and the associated experiments will be brought together with those of the previous chapter in Section 4.3 to provide a full picture of the work.

4.1 Research Framing

The previous chapter laid out the methodology for evaluating phenomena criticality for simplified models of a system in the context of transference of reinforcement learning poli-

cies from these simplified models to a referent system. This was meant to address limitations from previous methods found in the literature that do not evaluate the impacts of individual phenomena on the transference of policies learned in simulation. However, it did not explicitly give rise to how this information should be leveraged in actually developing these simplified models. This mirrors the gap in existing research discussed in Chapter 2 around general model development. Methods that exist in the literature are often ad-hoc and are targeted at specific systems. However, they have largely not found applications in developing models for training autonomous systems. It is expected that the phenomena criticality measures as defined in the previous chapter will be useful for this, but there still needs to be a directed strategy for developing these system models. This leads to the following research question:

Research Question 2.0: *Given appropriate measures of phenomena criticalities for a referent model, how should simplified models be constructed to achieve the greatest transference with the lowest complexity?*

This is a natural extension of the method defined in the previous chapter. That is, the effectiveness of any criticality measure can only be evaluated in the context of actually producing simplified models. However, this effectiveness is dependent on the eventual method implemented for using this information. This tight coupling makes it difficult to disentangle the effects of the metric itself from the development method. As such, it is important to consider the combination of the two as a whole.

While this is a critical question that has been raised throughout the modeling and simulation literature, there still remain few satisfying answers. As was discussed in Section 2.3, many existing methods are primarily focused on post-hoc verification and validation of an existing model with little work providing insight into how the models can be improved. In cases where there are better defined methods for evaluating and comparing models, such as in semi-conductor manufacturing, the methods are often highly specific or ad-hoc in nature.

[95, 127]

When investigating the literature in Section 2.3, it was hypothesized that much of this ad-hoc nature of model development is due to a lack of understanding of how lower-level modeling decisions affect high-level transference results. With the criticality measure defined in the previous chapter filling this role, it is reasonable to expect that simple development strategies can be used to leverage this new information and show improvement over more naive approaches that lack information on the relative importance of different phenomena. One of the simplest approaches possible would be to simply develop the models by including phenomena in decreasing order of criticality until a sufficient level of transference is achieved. This was employed in the example used throughout the previous chapter, and leads to the following hypothesis in answer of Research Question 2.0:

Hypothesis 2.0: *If phenomena criticality is measured as proposed and models are developed by including the phenomena in descending order of criticality, then the produced simplified models will show similar or greater levels of transference with fewer phenemena represented.*

To be more explicit, when a new model for a system is desired, the first step is to evaluate the criticality of each potential phenomena that has been identified. This will yield a ranking of each phenomenon from most critical to least critical. Given this ranking, we can now develop a series of increasingly complex models. First, we will include only the most critical phenomenon of interest. That is, given a base model of a system, such as a simple kinematic model with no dynamics considered, we add the effects of the most critical phenomena. We can then evaluate this model (if it was not included in the initial sample of models) and verify whether it is sufficient for our needs. If it is not sufficient, we can now add the second most critical phenomena, again evaluating this new model and verifying its sufficiency. This process is repeated until a sufficient model is identified.

This method of model development is proposed as a way to limit the need for addi-

tional exploration and optimization of phenomena once the criticality metrics have been measured. While simple, this is likely to work well as much of the information that would be gained through further optimization at this stage should be incorporated into the criticality measure itself. That is, because the criticality measure proposed in the previous chapter was evaluated by sampling models that included many combinations of additional phenomena, these coupling effects should at least be partially embedded in the evaluation for a given phenomenon.

This method also follows much of the main spirit of continuous iteration seen in the ad-hoc methods proposed in the modeling and simulation literature. The major improvement comes from using the defined metric to more intelligently develop each succeeding model. This is not to say that methods for verification and validation will no longer be necessary, but it is expected that models produced through this method will achieve consistent transference, and so be validated, earlier than those developed through a less informed approach.

Given the criticality measure developed in the previous chapter and the simple model development methodology above, a reasonable path forward for developing simplifications of a referent model has been laid out. However, there is still a major hurdle when it comes to applying this to a realistic system. To this point, it has been implicitly assumed that the referent system and the true system are one and the same. That is, by evaluating transference from the simplified models to the referent system we gain the same information as if we were transferring policies directly to the truth system. However, much of the motivation for developing simplified simulations comes from difficulties in obtaining data from a real system, suggesting this referent system will need to be a model of the true system of interest. That, and the fact that much of the sim-to-real literature discussed difficulties of transference when even relatively high fidelity simulations were used suggests that even these criticality measures may diverge if the referent model is not sufficient. This leads to the final research question to be addressed by this work:

Research Question 2.1: *How sensitive are these phenomena criticality measures and the resulting model development strategy to the fidelity of the referent model?*

This is a critical question to answer for this work to be useful in anything more than a theoretical sense. That is, if this methodology is so sensitive to the fidelity of the referent model that any small deviation results in invalid comparisons between phenomena, the value of the work becomes more limited.

To illustrate this point a bit more obviously, consider three systems that are considered under this method: the targeted truth system, the referent system model used to evaluate criticality, and the simplified models constructed through this methodology. Simplifying the notion of fidelity to a single scalar quality, we can consider the truth system at perfect fidelity, as it ***is*** the system we are trying to represent. Any behavior it exhibits should be considered truth.[1] At some lower level along this fidelity spectrum we find the simplified models developed through this methodology. Somewhere between these two points lies the referent system. This can be seen in visually in Figure 4.1.

This question is then asking how close to the truth system along this line of fidelity does the referent system need to be for evaluations of phenomena criticality between the simplified models and the referent system to still yield useful information on the true criticality measures of the same phenomena for transferring policies from the simplified models directly to the truth system. That is, how does the correlation between phenomena criticality measures evaluated with respect to transference to the referent system and measures evaluated with respect to transference to the true system change as the fidelity of the referent is changed? This is a difficult question to answer, and is part of why modeling and simulation remains as much art as science. Transference can be seen as an emergent property, relying on the interactions of many phenomena that have either been modeled or omitted.

[1] It should be noted here that even if there is data that has been collected from the true system, it is reasonable to continue to view this as only a model of the true data. This has been the argument of some in the modeling and simulation community, such as in [100], as the data is subject to the fidelity of any measuring devices, and can only be used to represent a necessarily finite portion of the true system's behavioral space.

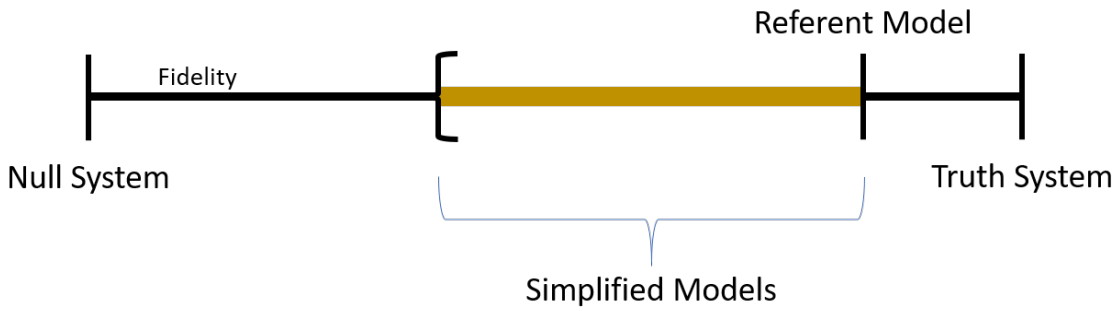


Figure 4.1: A notional representation of a practical implementation of this method. The truth model represents the actual system the policy is being trained to be used on. The referent system acts as the basis for the evaluation of phenomena criticality. The simplified systems are those models developed through this methodology. An underlying assumption of this method is that phenomena criticality measured by transferring from the simplified models to the referent model maintains similarity to the theoretical measures from the simplified models to the truth system.

Emergent properties such of this exhibit complex relations that can be hard to predict and evaluate consistently.

This complexity makes constructing general statements on fidelity requirements for referent models a fool's errand with our current tools. The ever-present hazard of unknown phenomena that affect the transference of policies between systems makes even statements on specific system classes difficult. However, if we take a slightly different view and a slight logical leap, we can at least recognize where this reliance on a referent can get us into trouble.

Referring back to Figure 4.1, we can take the final transference, from the simplified models to the truth system, as a nested transference. That is, the policy is first transferred from the simplified model to the referent model. Then this policy is again transferred from the referent model to the truth system. While this additional step isn't necessary in practice and is largely philosophical, it is a useful framing to consider when transference to the referent will no longer provide useful information. If the two part chain is broken, why would we expect the single step chain to be successful? This insight leads to the following hypothesis regarding Research Question 2.1:

Hypothesis 2.1: *If the referent model itself does not show reasonable transference to the true system, then the relative criticality measures evaluated with respect to it will not hold when evaluated on the true system.*

One thing to note of this hypothesis is that it is meant in general. The lack of transference between a referent and the true system may not preclude transference between a further simplified model and the true system. For a simple case of this, consider a referent that includes a so-called distracting phenomenon. That is, it includes a phenomena that is expected, but not present in the true system. Depending on the strength of this distraction, it may result in a lack of transference. A simplified model that omits this distracting phenomenon may then show transference. A similar possibility would be a truth system where two phenomena are tightly coupled in a balancing manner, but only one of these is considered in developing the referent. The lack of the second balancing phenomena may make the transference worse than if neither phenomena were included.[2] While either of these cases would likely signal a weakness in development of the referent, where the solution to this problem is considered out of scope for this work, it illustrates some of the weakness in this hypothesis. Even with this, this hypothesis provides a useful rule to signal danger when attempting to further simplify a referent system.

4.2 Experimental Definition

The previous chapter discussed the development of a methodology for the measurement of the importance of distinct phenomena to be used in simplified models of a referent. This importance is with respect to the transference of a reinforcement learning based policy from these simplified models to the referent system. This chapter has now laid out how these measures can be used in developing simpler models, and discussed some of the practical

[2]For a simple, though somewhat ridiculous example, consider a boat. The buoyancy of the water and gravity are coupled, balancing phenomena. If only one were included in the referent, the results would be disastrous, but omitting both phenomena and assuming the boat is held constant in this dimension could produce meaningful results.

implications to be considered when implementing this method. The previous section laid out the primary research questions and hypotheses that were formulated with respect to this method. The current section will now develop experiments to evaluate these hypotheses.

As was noted in Section 3.3, in reference to evaluating the phenomena criticality method, this method cannot be evaluated in a vacuum. The eventual output criticalities from the measurement method must be used in conjunction with the development strategy laid out above to develop simulation models of increasing complexity. The transference from these models can then be evaluated and should show improved transference relative to models of similar complexity whose phenomena were chosen through alternative methods. This is the basic procedure underlying each of the following experiments.

4.2.1 Experiment 1 Revisited: General Proof of Concept

The goal of this experiment is to provide an initial evaluation of the usefulness of the proposed criticality method in the context of developing simplified models of a referent system. This evaluation is in response to the two primary questions and hypotheses of this thesis. Those were *How can the criticality of potential phenomena for a simulation model be measured?* and *How should these phenomena criticality measures be leveraged in developing simplified system models?* The previous chapter discussed a sampling-based methodology for developing these criticality measures. The previous sections of the current chapter laid out how these measures should be leveraged.

As was discussed in Section 3.3.1, this first experiment will take the form of evaluating control of dynamical systems. The policy to be trained will be the controller, the measure of success for binary transference will be stability of the trained controller, and the performance of the policy will be measured as a linear quadratic cost of the states and control inputs enforced by the controller.

Given the model development strategy defined in Section 4.1 above, we can now get more specific on how this will be evaluated. First, the phenomena criticality measurement

method will be applied to a given referent system. This will result in the output of a set of criticality measures, one for each phenomena. Then, following the strategy for model development outlined above, we can define a set of simplified models of increasing complexity.

To illustrate this, we can continue the example used previously, where we were considering a system with four distinct phenomena. At the conclusion of the phenomena criticality measurements, we found the second phenomenon to be the most critical, the fourth phenomenon to be second most critical, the third phenomenon to be the third most critical, and the first phenomenon to be the least critical. Now, according the the model development strategy, we can define a set of simplified models of increasing complexity by including phenomena in order of decreasing criticality. Ignoring the null model with none of the phenomena included, these are defined by the strings 0100, 0101, 0111, and 1111. The inclusion of the final "simplified" model is included only for clarity, as this represents the full referent and is therefore unnecessary.

For this experiment, we will implement each of these models and evaluate the binary, performance, and potential transference metrics, defined in Equation (3.1), Equation (3.2), and Equation (3.5), respectively, for each model. We can then plot these measures relative to the fidelity[3] of the associated models. So, continuing the example, say model 0100 does not transfer successfully, so neither performance or potential transference have a meaningful interpretation. Continuing to the next model, we already have that model 0101 will transfer successfully, but now we have that it achieves a performance transference of 0.5 and a potential transference of 0.75. Model 0111 is then tested, it transfers successfully, and has a performance transference of 0.25, and a potential transference of 0.2. The referent model obviously transfers, and has performance and potential transference metrics of 0.0. This produces the following transference-fidelity curves:

To simplify this evaluation, these curves can be converted into a single metric by cal-

[3] Again, taken here to simply mean the proportion of phenomena from the referent system included.

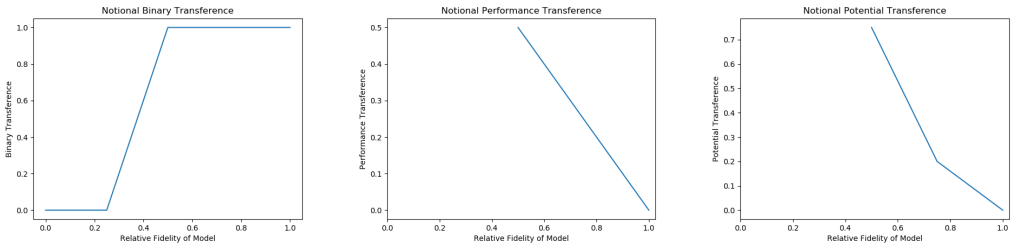


Figure 4.2: Notional representations of the transference vs fidelity graphs used to tests the phenomena criticality evaluation method.

culating the areas under these curves. For binary transference, the ideal measure for this metric would be 1.0. This would imply that the models produced by the simplification development methodology produces models that transfer to the referent as soon as the most important phenomena is included. Clearly, this metric is system dependent, as some systems will likely present a greater challenge to transference than others. Even in this case, comparison between methods using this metric is meaningful as the greater this measure, the greater the transference achieved at lower complexity levels.

This measure is also useful in evaluating performance and potential transference. However, as these two measures are related to the difference between the simulated policy and policies evaluated or trained in the real world, the goal is to now minimize them. That is, the ideal simulation model would achieve both perfect prediction and full training with the fewest phenomena accounted for. This would yield an area under the curves of 0. This ideal case is unlikely to happen in the real world though, so comparisons of different methods will largely come down to which yielded a lower value.

The above defined how a given set of phenomena criticality measures can be leveraged by using them to develop a set of models of increasing complexity. While the proposed method for identifying critical phenomena is likely to show positive results, it must be evaluated in context. That is, simply finding a useful ordering of criticality is not sufficient. It must find a better ordering of phenomena than a naive method while also being more efficient than a brute force method. Similarly, it should compare positively in this tradeoff

against other simple heuristics that can provide a proxy for ad-hoc methods. As such, this section will establish three distinct baselines for comparison.

Naive Baseline

The first baseline is the simplest, a fully naive approach. For this baseline, phenomena to be included at a given model fidelity level will be randomly selected according to a uniform distribution. This mimics the case where nothing about the interactions within the system is known. As such, it can be considered the null hypothesis for this experiment, where no useful information exists to extract. While this is a useful baseline to compare against, ensuring that valuable information is actually identified, it is not necessarily indicative of current best practices.

Heuristic Baseline

The second baseline to compare to will be a simple heuristic baseline. Some simple heuristic on the phenomena will be used to determine the expected importance of each phenomena prior to any experimentation. This heuristic will be dependent on the actual system being considered, but can be thought of as a proxy for evaluation by a subject matter expert. This method requires no experimentation, and may produce useful results if a reasonable heuristic can be found. However, it should be noted that the accuracy of these sorts of heuristics will likely be inversely correlated with system complexity, and so becomes less useful for more realistic systems.

Greedy Baseline

The third, and final, baseline to be compared to will be a greedy search method. This will use a full factorial search of the simplification space to find the simplest model that achieves binary transference. From this point on, model complexity will be added one phenomenon at a time. To do this, each remaining phenomena at a given stage will be added to the model

individually and evaluated. The phenomena that improves the model the greatest, defined as maintaining binary transference with ties broken through performance transference, will be added to the model. This process will then repeat. In this way, this is a quasi-full factorial method that largely mimics the current practices of the modeling community.

For clarity with regards to the third baseline, consider our four phenomena system from earlier. The first stage of this baseline will first evaluate every possible single phenomenon simplified model for transference. That is, 1000, 0100, 0010, and 0001. If none of these models achieve binary transference, then all two phenomena models will be evaluated. That is, 1100, 1010, etc. Assuming multiple of these two-phenomena models achieve successful binary transference, the tie will be broken by performance transference. Say this leads to the model 0101 being selected. From here, we evaluate all three phenomena models conditioned on them containing the second and fourth simplifications. That is, we evaluate models 1101 and 0111. We select whichever shows the best transference as defined above as the third model in our simplification set.

This is an attractive baseline to compare to as it will by definition find the minimal set of phenomena that leads to transference, while mimicking the current ad-hoc nature of modeling that often follows a trial and error process. Similarly, if the first transference occurs with relatively simple models, this could be a fairly inexpensive method.

While attractive in theory and simple in implementation, this method has significant flaws. First, there isn't a clear method for breaking ties. For example, what if no single phenomena allows for any transference? In this case, two options are immediately apparent. First, the phenomena to carry forward can be selected at random. While simple, this will almost surely lead to non-optimal simplification selections as the number of considered phenomena grows.[4] The second alternative, which was described above, would be to

[4]While a formal proof isn't given here, consider the following sketch. Assuming there is an appropriate ordering of phenomena, the likelihood of selecting the correct phenomena at any given stage, conditioned on the previous phenomena being correctly selected, is $\frac{1}{N-n}$ where N represents the total number of phenomena considered and n representing the number of previously selected phenomena. Therefore the probability of naively choosing the M correct phenomena among a total set of N phenomena is $\prod_{i=0}^{M-1} \frac{1}{N-i}$, which tends towards 0 as either N or M increases.

consider all combinations of phenomena for increasing phenomena levels until a set that produces transference is identified. This is reasonable for small systems, but faces combinatrorial growth both in the number of phenomena considered and the initial fidelity level required to differentiate between simplifications. This growth could quickly offset the gains in model simplicity.

With the Naive, Heuristic, and Greedy baseline methods defined above, each method will be implemented on a system to identify a phenomena ranking. This includes the method proposed in the previous chapter to be used in fulfilling Hypothesis 1.0 and Hypothesis 2.0 above. These rankings will be used to develop a set of simplified models of increasing complexity according to the ranking. The transference curves will be defined, and each method will be compared. The method proposed throughout this and the previous chapter must at a minimum improve upon the naive method to be considered successful. Similarly, though there may be problems in defining heuristics for more complex systems, it must improve upon the heuristic method for simple systems for Hypotheses 1.0 and 2.0 to be considered supported. Finally, it must also approach the results of the Greedy Baseline at reduced computational cost to show its worth.

4.2.2 Experiment 4: Effect of Referent Fidelity

The previous section laid out a full proof of concept study to show the methodology for measuring phenomena criticality laid out in the previous chapter, when paired with a simple model development strategy laid out in the beginning of this chapter has worth. Experiments 2 and 3 discussed in the previous chapter laid out how different pieces of the method for measuring phenomena criticality will be evaluated. Specifically, those were the strategy used to sample the simplified models to evaluate for this measure, and the underlying transference metric to use in the evaluation of the different sampled models. Those experiments will follow a similar pattern, where they will implement the full methodology and compare

the effects of these two decisions on the areas under the fidelity-transference curves.

This chapter then looked at some of the practical considerations that must be made when considering the problem of transference. Namely, access to the true model may be infeasible or prohibitively expensive. As such, it may be unreasonable to evaluate transference directly. Instead, the method must use a referent model that is explicitly a simplified model of the true system. This led to Research Question 2.1, which asked how sensitive this whole methodology for model development would be to the fidelity of the referent model. This led Hypothesis 2.1, which posed that if the referent model cannot produce transferable policies, it also cannot be used for evaluating phenomena criticality.

In evaluating this question and hypothesis, the first step must necessarily be to create a distinction between the truth model of the system and the referent model used in evaluating phenomena criticality. For this, we will look at the case where the referent model is a pure simplification of the truth model. That is, we will not be considering the case where the referent includes so called distractor phenomena as was discussed as a possible case in Section 4.1.

While this reduces some of the generality of the results of this experiment, the difficulty in defining distractor phenomena that are both significant enough to be noticeable while creating a sufficiently small change in behavior that training can be completed successfully is a very difficult problem in its own right. This is similar to a developing field of machine learning called "adversarial training", where examples are designed specifically to fool machine learning trained policies. [31] The difficulty in creating these examples likely means the intentional creation of distractor phenomena would be too great to be feasible for this experimentation. Also, even with this restriction, there is still a lot of valuable information to be gained about this method. Looking purely at simplified models can still stress the evaluation method, and will likely find its breaking points. That is the goal of testing this hypothesis.

So, given a referent that is a pure simplification of the truth model, we now can imple-

ment the method on this simplified referent. This method will produce a set of phenomena criticality measures. Given that the truth system is simple enough for access, we can then conduct the same analysis using the actual truth system itself as the referent. This will give us a second set of phenomena criticality measures. Considering only those phenomena that were present in both the true system and the simplified referent, we can then evaluate the correlation between these two. This can be done with either the raw criticality scores or as a comparison of the rankings.

These correlations will be used to evaluate whether the criticality measures from simplified referent hold with respect to the true system. The goal to test the hypothesis then will be to identify simplified referents that do not transfer to the true system. If the criticality measure from these referent systems correlate with the true measures significantly worse than measures derived from referent systems that do transfer to the referent system, the hypothesis will be supported and the warning will be considered valid.

4.2.3 Experiment 5: Practical Case Study

To this point, 4 separate experiments have been laid out in response to the research framework that has been proposed. These are:

- Experiment 1: Evaluation of the methodology as a proof of concept
- Experiment 2: Evaluation of the effects of sampling distribution
- Experiment 3: Evaluation of the effects of evaluation metrics
- Experiment 4: Evaluation of the effects of referent fidelity

While these experiments are valuable in determining the potential usefulness of this method for comparing phenomena for developing simplified models of a system, they are relatively heavy evaluations that will implicitly require a simple system. This is because more realistic systems would be too expensive to experiment sufficiently for any of these

questions to be properly evaluated. As such, a final experiment will be needed to evaluate the full methodology on a realistic system.

This will serve two main purposes. First, it will provide a venue for lessons learned through Experiments 2, 3 and 4 to be integrated into the methodology. Second, it will provide evidence that the resulting criticality measures produced by this method are also meaningful for a more complex system. This experiment will follow the same format as Experiment 1. That is, a truth system will be defined, with a referent model defined based on this truth system. The criticality evaluation method as described Evaluating Phenomena Criticality will be implemented to determine the relative importance of each phenomena in this referent. These rankings will then be used to develop increasingly complex models of the true system, and the transference of reinforcement learning policies from these simplifications will be evaluated. If the resulting transference curves are qualitatively similar to those curves defined during Experiment 1, this will be considered successful and provided as evidence of the method's usefulness.

4.3 Summary

The previous chapter, Evaluating Phenomena Criticality, discussed the proposed method for measuring phenomena criticality within the context of a simplified model of an autonomous system. The current chapter then focused on developing a simple model development strategy to leverage these measures to construct simpler models that still show adequate levels of transference. This was framed according to Research Question 2.0, which asked how the criticality measures should be leveraged when developing simplified models of a system for simulation-based training. Section 4.1 developed Hypothesis 2.0 in response to this question. This posited that if a set of models of increasing complexity were developed by including phenomena in order of decreasing criticality the simplified models within that set would show similar or greater levels of transference with fewer phenomena represented.

Having this foundation for developing simplified models, some considerations for applying this to practical autonomous systems were discussed. Chief among these was the potential disconnect between the true targeted system for policy deployment and the referent model used in the evaluation of phenomena criticality. This was framed by Research Question 2.1. Hypothesis 2.1 answered this question by suggesting that if the referent model used to evaluate phenomena criticality was not capable of training transferable policies itself, it would also be incapable of measuring phenomena criticality accurately.

Given this framework for investigating leveraging the measured phenomena criticalities, experiments to evaluate these hypotheses were developed. As was noted, these experiments can not be evaluated in a vacuum, and must be incorporated with the experiments laid out in response to the hypotheses proposed in the previous chapter. In this way, the proof of concept experiment presented for Hypothesis 1.0 in Chapter 3 was developed further to add a comparison to representative baseline methods. These baselines were developed to represent a Naive method, a Heuristic method, and a Greedy method. Comparison to these methods will allow for an evaluation of Hypothesis 2.0. An additional experiment to evaluate the effects of the fidelity of the referent model was described in response to Hypothesis 2.1. Additionally, a final case study to integrate expected lessons learned from Experiments 1-4 was described to evaluate if this method can be applied to more practical systems. Further details on the differences between these two experimental systems, the simple class used for Experiments 1-4 and the realistic class used for Experiment 5, will be provided in the next chapter. These experiments and their aims are summarized in Table 4.1.

With a framework for the research now set, the next chapter will discuss the specific implementation details and results of the experimental framework. This will include discussing these results in the context of support or refutation of these proposed hypotheses. Additional discussion on surprising results, possible further interpretations, and follow on questions will be discussed as relevant.

Table 4.1: Summary of experiments to evaluate research questions and hypotheses regarding phenomena criticality measurement

Experiment Number	Experiment Name	Data Gathered	Hypotheses Tested
1	Proof of Concept	Comparison of transference curves against baseline methods	Hypothesis 1.0 Hypothesis 2.0
2	Effect of Sampling Distribution	Comparison of transference curves for models whose phenomena criticality is derived from the method using different simplification sampling strategies	Hypothesis 1.1
3	Effect of Scoring Metric	Comparison of transference curves for models whose phenomena criticality is derived from the method using different scoring metrics Comparison of phenomena rankings as the referent model is simplified	Hypothesis 1.2
4	Effect of Referent Fidelity	Comparison of transference curves for models whose phenomena criticality is derived from the method applied to simplified referents	Hypothesis 2.1
5	Practical Case Study	Evaluation of transference curves for a more complex system	Hypothesis 1.0 Hypothesis 2.0

CHAPTER 5
EXPERIMENTAL RESULTS

This work has set out to study the effects of modeling choices on the transference of policies learned through reinforcement learning in simplified simulations of a system. This has been framed through measuring the importance of different phenomena to be included in the simulation model, here called phenomena criticality. Evaluating Phenomena Criticality laid out how this measurement can be approached, as well as primary research questions and hypotheses to evaluate this proposed method. Developing Simpler Models then takes this criticality measurement and pairs it with a simple strategy to develop simplified models of a system. Again this chapter laid out associated research questions and hypotheses that were meant to evaluate this pairing.

These chapters also provided high level designs for experiments to evaluate these questions and hypotheses. This chapter will now focus on the specific implementations for these experiments as well as the results of these implementations. First, the two main systems used in these experimental analysis are discussed. Then, the results from each of five major experiments are discussed in detail. This discussion will primarily focus on the support or refutation of the hypotheses presented in the previous two chapters, though some broader discussion will be included as warranted.

5.1 Experimental Systems

Two main classes of systems were considered for these experiments. The first was linear dynamical systems. This was used for early experimentation due to their simplicity and ease of simulation. This allowed for the method to be rapidly tested, iterated and improved. In addition, linearized representations are often used for more complex systems so they have significant meaning for many control tasks. However, nearly all systems of interest

for robotics are nonlinear in nature. As such, the second system considered was a simple nonlinear system, called the Acrobot. [108] This system is defined as a double pendulum with a motor between the two links for applying control torque. While simple, this system represents many problems that are common in robotics: nonlinear systems, underactuation, and chaotic uncontrolled behavior. These two types of systems are discussed in further detail below.

5.1.1 Linear Systems

Consider a linear system of the form:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{5.1}$$

Much of classical control literature deals with the development of state feedback controllers for these types of systems, often of the form $\mathbf{u} = -K\mathbf{x}$. It is then common to try to design an optimal controller that minimizes some cost function for the system. One of the most common forms of cost function considered is the Linear Quadratic cost function:

$$\int \left(x^T Q x + u^T R u\right) dt \tag{5.2}$$

This cost function has two parts, a state cost that penalizes the system for staying away from the equilibrium point at the origin, and a cost that penalizes the system as it applies additional control to bring the system towards the equilibrium point. The Q and R matrices are weighting matrices, allowing a designer to give different importance to different states or controls to reflect the practical considerations for the system. While analytically derived controllers exist for these systems with this cost function, it is also reasonable to apply reinforcement learning frameworks to this problem to further understand how they work. It is also asserted then that we can learn how different modeling choices affect the transference of learned policies by investigated linear systems.

In order to do this, we must understand what constitutes a "phenomenon" for a linear system. As was discussed in Background and Related Work, the behavior of any given system is determined by the phenomena that are present within the system itself and their interaction with the environment. For realistic systems, a simulation designer will need to decide on which phenomena to include to ensure accuracy and which to exclude due to computational constraints. If we squint, we can see that each element of the given A and B matrices in Equation (5.1) can be considered a different phenomenon for the system. That is, each element describes the impact of one state or control input on a resulting state derivative for that system, and therefore describes how each state or input affects the behavior of the system. As such, these can be considered an analogue to more complicated phenomena for other systems. By setting any given element of the A or B matrix to zero, we have essentially simplified the model to ignore that phenomena. While this admittedly has little impact on the computational costs of simulating this system, it is analogous to removing a potentially expensive to compute phenomena from another system. This simplified model will be represented by the following system:

$$\dot{\mathbf{x}} = \hat{A}\mathbf{x} + \hat{B}\mathbf{u} \tag{5.3}$$

To further illustrate this simplification, we can look at an example system. Say we have a two state, two input system. We then have:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} ; \; B = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \tag{5.4}$$

This gives us 8 possible phenomena to included in our model of the system, and 256 possible simplified models. Using the same strategy as used throughout the example given in Chapter 3 and Chapter 4, we can encode each of these possible simplified models as an 8 character binary string. The string 11111111 would represent the truth model, while the string 00000000 would represent the null model. As was mentioned in the previous

chapters, the ordering of the phenomena at this point is purely arbitrary. To convert from string indices to element position, we'll flatten each matrix by concatenating each row from top to bottom into a single row vector, then concatenate the row vectors representing the A and B matrices respectively. So, the above system would yield the phenomena list of $(a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}, b_{1,1}, b_{1,2}, b_{2,1}, b_{2,2})$. So, the simplification string 10011110 would represent the simplified system defined by:

$$\hat{A} = \begin{bmatrix} a_{1,1} & 0 \\ 0 & a_{2,2} \end{bmatrix} ; \hat{B} = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & 0 \end{bmatrix} \tag{5.5}$$

For simplicity, the truth systems represented in Equation (5.1) and Equation (5.4) will be referred to as M, the simplified model represented in Equation (5.3) and Equation (5.5) will be referred to as $\hat{M}$. The goal of sim-to-real would then be to train a policy, π on the simplified system, $\hat{M}$ that is applicable in the true environment, M.

Because the true system is virtual in this case, this transference can be directly measured. Additionally, the number of phenomena to consider is finite for these linear systems and a full factorial of the simulation design space can be considered for small systems. These are both very attractive features for initial experimentation. They also lend well to more in depth study of the effects of different aspects of the methodology as outlined in Evaluating Phenomena Criticality and Developing Simpler Models. As such, linear systems will be used for the initial proof of concept, Experiment 1, as well as Experiments 2-4 that will further probe the limits of this method.

There is an important note to make here though. While the goal of this work is to investigate the impacts of modeling on reinforcement learning, this is an incredibly time consuming process. Continuous control problems such as this are still difficult for many reinforcement learning frameworks, and each system may take many hours to learn a suitable controller. For these initial experiments, it is important to remember that reinforcement learning is above all else a framing for optimization. That is, the goal is to produce some

sort of optimal controller. As was mentioned above, there are already analytically derived optimal controllers for these systems under these cost functions, specifically LQR control. Therefore, it is reasonable to expect that any properly functioning reinforcement learning framework will closely approximate the behavior produced by these analytically derived controllers given sufficient training time. So, by evaluating the effects of transferring LQR derived controllers we should closely approximate the effects of transference on reinforcement learning based policies for this system.

This assertion allows for a significant reduction in evaluation time, and will therefore be used for Experiments 1-4 to allow for greater numbers of systems to be considered. In order to ensure these results still hold for policies that are truly trained through reinforcement learning, Experiment 5 will use fully trained policies in its final evaluation of transference curves.

5.1.2 Acrobot System

The acrobot system has been described in previous literature as a simple method for understanding common problems in the control of robotic systems. [108, 112] It has many attractive features, including nonlinearities, underactuation, and chaotic uncontrolled behavior. These features arise form a relatively simple system, shown in Figure 5.1. This is a simple double pendulum, with a single control torque input at the joint between the two pendulum links. The goal of this system has been defined in many ways in the literature. However, for simplicity we will use the case of raising the end point of the second link to an altitude of at least one link length above to fixed joint.

This system was slightly altered from that used in both [108] and [112] to add additional phenomena to be considered. First, rotational springs were added to each joint. These springs are positioned such that they are relaxed when the two links are pointing downwards. This has two major effects. First, it adds a linear disturbing force that alters the unstable equilibrium position from directly upright to slightly tilted. Second, it adds a

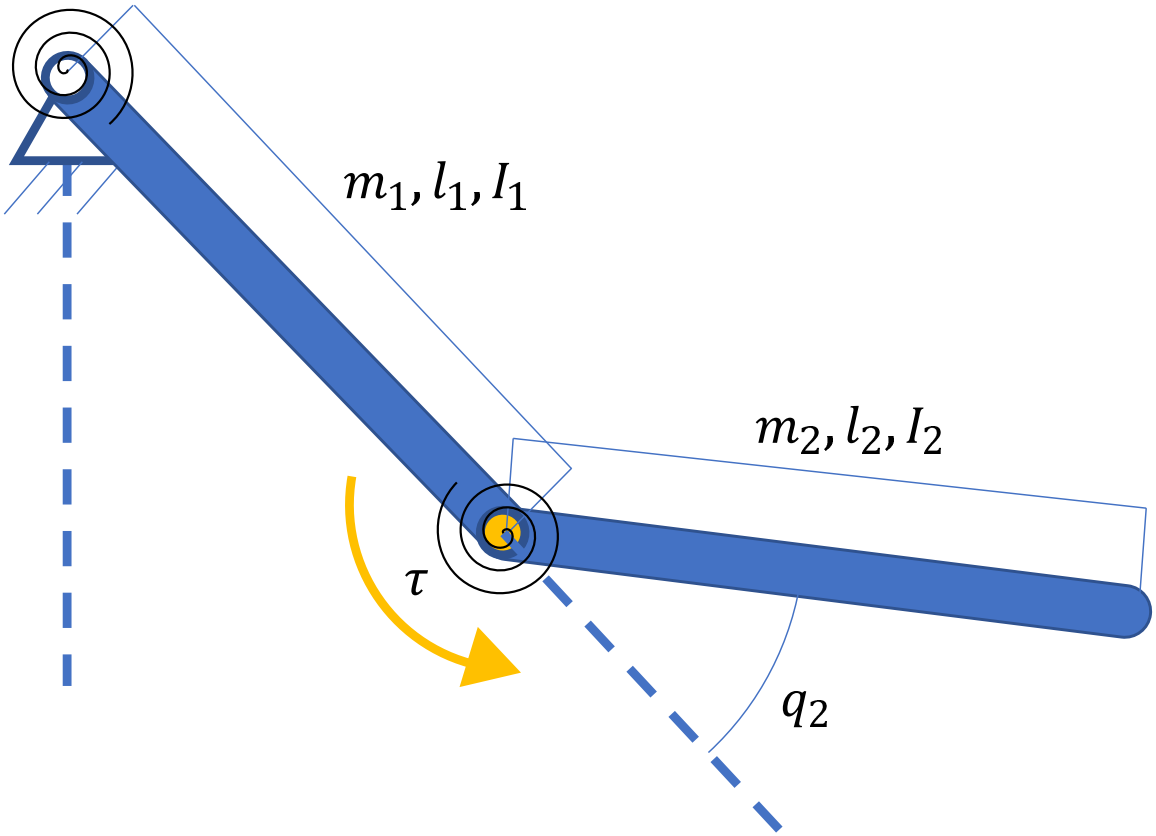


Figure 5.1: Illustration of the Acrobot system, similar to that described in [112]

stabilizing force for the second link, somewhat simplifying the problem. These can be seen as a pair of phenomena, one (the spring between the two links) making the problem slightly easier, and one (the spring attached at the first link) making the problem slightly harder.

A second alteration is the addition of damping torques acting on the joints (not shown in Figure 5.1). These torques take both a linear and quadratic form with respect to the angular velocity at the joint.The linear term can be considered a simple representation of friction at the joint, while the quadratic term can be considered a simple representation of air resistance.

The final alterations deal with the application of torque from the motor. The system as described in [112] is limited to three distinct torque settings: -1, 0, and +1. While interesting from a reinforcement learning perspective, and useful for applying the popular Deep Q-Learning technique from [78], this is limiting. The description from [108] describes a

continuous and unlimited supply of torque from the motor. While less limiting, this now becomes unrealistic. So, the system used for this experimentation will consider a continuously controllable torque application with limits. For simplicity, these limits will simply be be applied as a clipping of the requested torque if they are beyond the capabilities of the motor. Additionally, the torque will be considered noisy, with bounded white noise applied to during the application of torque.

Given these alterations, the governing equations for the system are as given below:

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = 0 \tag{5.6}$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = \tau \tag{5.7}$$

Where

$$\begin{aligned}
d_{11} &= m_1 l_{1,c}^2 + m_2\left(l_1^2 + l_{2,c}^2 + 2l_1 l_{2,c}\cos q_2\right) + I_1 + I_2 \\
d_{22} &= m_2 l_{2,c}^2 + I_2 \\
d_{12} &= m_2\left(l_{2,c}^2 + l_1 l_{2,c}\cos q_2\right) + I_2 \\
d_{21} &= d_{12} \\
h_1 &= c_{1,1}\dot{q}_1 + c_{1,2}\dot{q}_1\left|\dot{q}_1\right| - m_2 l_1 l_{2,c}\sin q_2 \dot{q}^2 - 2m_2 l_1 l_{2,c}\sin q_2 \dot{q}_1 \dot{q}_2 \\
h_2 &= c_{2,1}\dot{q}_2 + c_{2,2}\dot{q}_2\left|\dot{q}_2\right| + m_2 l_1 l_{2,c}\sin q_2 \dot{q}_1^2 \\
\phi_1 &= k_1 q_1 + \left(m_1 l_{1,c} + m_2 l_1\right) g\sin(q_1) + m_2 l_{2,c} g\sin(q_1 + q_2) \\
\phi_2 &= k_2 q_2 + m_2 l_{2,c} g\sin(q_1 + q_2)
\end{aligned}$$

For these equations, $c_{i,j}$ represents the damping coefficient for joint i of the j^{th} order damping, k_i represents the spring constant for the spring located at the i^{th} joint, and $l_{i,c}$ represents the center of mass for the i^{th} link.

This gives a number of phenomena that can be chosen to be included or omitted from a simplified model of this system used for training. Separating the damping torques both by linear vs quadratic damping, as well as joint location, and considering the application of gravitational forces as a modeling choice, this presents 9 phenomena of interest. These are:

1. The shoulder spring, omitted by setting $k_1 = 0$
2. The elbow spring, omitted by setting $k_2 = 0$
3. Linear damping at the shoulder joint, omitted by setting $c_{1,1} = 0$
4. Quadratic damping at the shoulder joint, omitted by setting $c_{1,2} = 0$
5. Linear damping at the elbow joint, omitted by setting $c_{2,1} = 0$
6. Quadratic damping at the elbow joint, omitted by setting $c_{2,2} = 0$
7. Gravitational forces, omitted by setting $g = 0$
8. Continuous vs discrete torque applications
9. Limited torque availability
10. Noisy torque application

As was discussed, it then becomes a modeling choice whether to include or omit each of these phenomena in the training model, yielding 1024 possible simplified models. While none of these phenomena in and of themselves are complex enough to significantly impact simulation speed, they are expected to play a similar role as more complex phenomena. As such, applying the method to identify the simplifications that can be made for this system will show whether or not it is realistic to expect it to provide meaningful results on more realistic systems. Additionally, it is clear these phenomena will not have equal contributions to transference. The omission of gravity will likely yield behaviors that do not

properly swing up the end on the real system, greatly impacting transference. Other factors, such as the quadratic damping terms, should have a much smaller effect on transference.

Even though this is a relatively simple system, it is still quite difficult from a reinforcement learning perspective. As such, this system will only be used in Experiment 5 as a final test to evaluate whether this method is applicable to more realistic systems than just the linear systems discussed above.

5.2 Experimental Results

The primary motivation behind this work is the difficulty of transferring reinforcement learning based policies from the simulation environments used to train them to the true systems in the real world. The two previous chapters laid out the major research framework when investigating this problem. Mainly, this was framed as a series of research questions and associated hypotheses, outlining the measurement and subsequent use of the criticality of distinct phenomena. In discussing these questions and hypotheses, five experiments were described. These are summarized in Table 5.1. The previous section discussed the systems to be used in evaluating these experiments. The linear systems will be used for Experiments 1-4, while the Acrobot system will be used to evaluate experiment 5. Each individual experiment is discussed in the context of the research framework. Additional discussion of interesting results will be included as allowed.

5.2.1 Experiment 1: Proof of Concept

The first proposed experiment was a simple proof of concept. It was laid out in detail in both Section 3.3.1 and Section 4.2.1. The goal of the experiment was to evaluate the two primary hypotheses driving this thesis. Those were Hypothesis 1.0 and Hypothesis 2.0, that a sampling-based approach to measuring phenomena tied with a simplistic model development strategy will yield simplified system models that can achieve similar levels of transference as compared to a referent model.

Table 5.1: Summary of experiments to evaluate research questions and hypotheses regarding phenomena criticality measurement

Experiment Number	Experiment Name	Data Gathered	Hypotheses Tested
1	Proof of Concept	Comparison of transference curves against baseline methods	Hypothesis 1.0 Hypothesis 2.0
2	Effect of Sampling Distribution	Comparison of transference curves for models whose phenomena criticality is derived from the method using different simplification sampling strategies	Hypothesis 1.1
3	Effect of Scoring Metric	Comparison of transference curves for models whose phenomena criticality is derived from the method using different scoring metrics Comparison of phenomena rankings as the referent model is simplified	Hypothesis 1.2
4	Effect of Referent Fidelity	Comparison of transference curves for models whose phenomena criticality is derived from the method applied to simplified referents	Hypothesis 2.1
5	Practical Case Study	Evaluation of transference curves for a more complex system	Hypothesis 1.0 Hypothesis 2.0

To conduct this evaluation, a nominal implementation of the phenomena criticality measurement method discussed in Hypothesis 1.0 and outlined in Section 3.2 was developed. Specific details on this implementation can be found in Section B.1. The simple strategy for model development based on these criticality measures discussed in Hypothesis 2.0 and outlined in Section 4.1 was also implemented. For this initial experiment, linear systems as laid out in Section 5.1.1 were used as the truth systems.

The results shown below were produced by considering a set of fifty randomly generated truth systems. As discussed above, each system was constructed by creating appropriate A and B matrices whose elements were drawn from a uniform distribution over the interval $[-1, 1]$. These systems will from here on be referred to as the "truth systems". These truth systems have two key features: they are almost surely controllable, while also almost

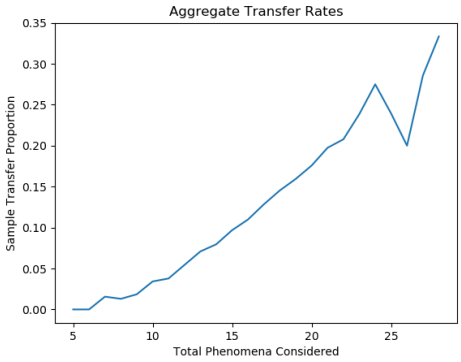
surely being unstable in the uncontrolled case. So, each system represents a non-trivial but possible stabilizing controller synthesis problem. The method was then implemented to measure the criticality of each element of these truth systems. These criticalities would then be used to define a set of models with increasing fidelity.[1] Transference for this set would then be compared to other sets of models produced by three baseline methods, as laid out in Section 4.2.1.

As discussed in in Section 3.2, the identification of critical phenomena for these simplified systems follows a four step process. The results for each of these four steps is visualized below in Figure 5.2. First, the simplification space of the truth system is sampled to produce an adequate number of simplifications to evaluate, as laid out in Section 3.2.1. For each simplification in this sample, a controller is synthesized, with a linear quadratic cost acting as a reward. This controller represents the simplified policy, π_S. As was laid out in Section 5.1.1 above, this controller was an analytically derived LQR controller to speed up the process for initial experimentation, as a well functioning reinforcement learning framework will approximate this controller given sufficient training time. The simplified policy is then applied to the truth system, and its stability properties tested. If the transferred controller is stabilizing, the transfer is considered successful.

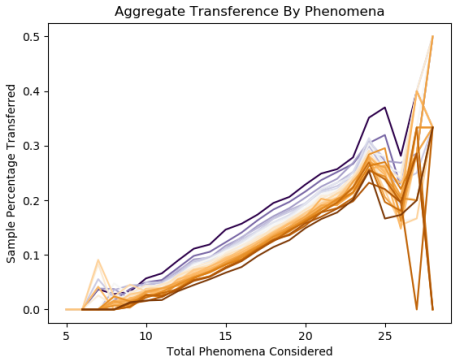
By analyzing many simplifications at a given fidelity level, a rough estimate of the expected transference rate for a given fidelity level can be calculated, as shown in Figure 5.2a. Clearly, these are not promising results for transference overall. Even at relatively high fidelity levels, the raw transference rates never rise above one in three. A second thing to note about Figure 5.2a is the simplification strategy behaves as expected, largely sampling from moderate fidelity simplifications. No systems that contained below 5 phenomena or above 29 phenomena were considered. The sampling distributions that were used are can be seen in additional detail in Section B.2.

The next step of the process classifies each simplification based on the inclusion or

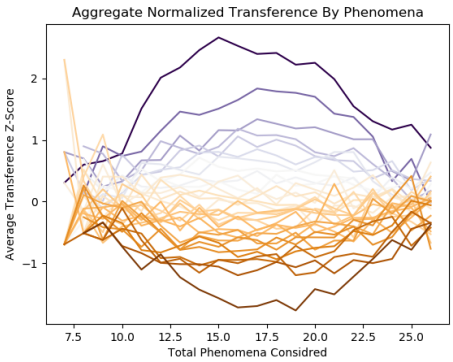
[1] As before, fidelity from here on will refer simply to the number of phenomena considered in the model.



(a) Aggregated Raw Transference



(b) Raw Transference By Phenomena



(c) Normalized Transference By Phenomena

Figure 5.2: The process of identifying critical phenomena for a model. First, the left figure shows the aggregate transference rates taken across all models of a given fidelity level. Then, the middle graph shows these aggregate separated into classes that all contain a given phenomena. Finally, the right graph shows these normalized for each fidelity level to identify critical phenomena across the range of sampled fidelities. For the middle and right graphs, each trace is colored according to the criticality of its characteristic phenomena for clarity.

omission of an individual phenomenon. As discussed in Section 3.2.2, these sets are defined according to:

$$\mu_i = \{m \in M : p_i \in m\} \tag{5.8}$$

That is, each set, μ_i, is defined as the collection of models, m, within the set of all sampled models, M, that contains the i^{th} phenomena, p_i. So, set one would contain all sampled simplifications that contain the first phenomena of the referent model. Set two

would contain all sampled simplifications that contain the second phenomena of the referent model, and so on. These classifications are not mutually exclusive, as any model with multiple phenomena present by definition exists in multiple classifications. However, by considering and comparing the aggregate affects of these models they do reveal information about contributions of each individual phenomena. This is similar to other approaches for identifying critical members, like ANOVA. The aggregate transference rates within a classification can then be calculated and compared across classifications, shown in Figure 5.2b. For this graph, each line represents the resulting transference rates for each classification denoted above. These results clearly show a stratification of transference rates, supporting the assumption that individual phenomena will account for differing amounts of transference ability.

While these results show an obvious stratification, they cannot be used automatically in quantifying the criticality of each individual phenomenon. This is because the resulting transfer rates are strongly correlated with fidelity level. If a phenomena was by chance included in more high fidelity simplifications, it would artificially increase its score. As such, a final step before ranking the phenomena is to normalize the returns by fidelity level. This is shown in Figure 5.2c, where the returns are scaled using the Standard Scaling approach, which calculates the Z-score of a given group, at each fidelity level to return a distribution over the model classes with zero mean and a standard deviation of one. This allows for a more representative comparison across a range of fidelities.The final step in the method is to give a single criticality measure for each phenomena. This is done by simply taking the mean across all fidelity levels.

There are two important features to note about this normalized figure. First, the separation between phenomena is most pronounced at the extremes of the criticality spectrum. That is, the difference between the most important phenomena is greater than the difference between moderately important phenomena. This further supports the assumption stated above, and lends evidence to the hypothesis that simplifications can be designed to achieve

greater transference even at low fidelity levels.

The second feature to note is the general trend of the spread of criticality. Ignoring the lowest fidelity simplifications, where there are few sampled simplifications and transference is rare, the spread in criticality decreases as the fidelity of the simplifications increases. This has a few possible interpretations, but a likely reason would be that it is an artifact of the inclusivity of the simplification classifications.As the fidelity of the simplifications grow, there becomes a larger overlap between the simplification classifications. Extracting the influence of a single phenomena becomes more difficult as these overlaps grow. This supports the choice of uniform sampling across the simplification space, even though this does not yield a uniform sampling of the fidelity space. This allows for phenomena influences to be more efficiently extracted and reasonable estimates of rankings to be produced with fewer samples.

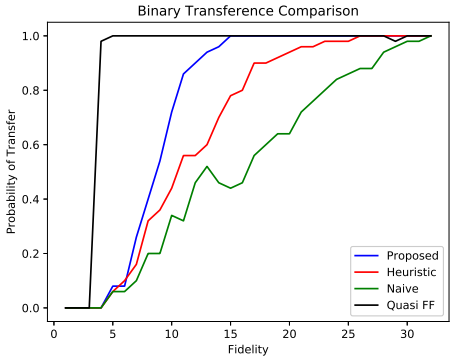
While these figures certainly support the idea of the existence and identification of critical phenomena, they don't reveal if these identifications are actually useful. To evaluate this, simplifications built according to Hypothesis 2.0 were constructed, policies for these simplifications were synthesized, and the transference of these policies to their truth systems was tested. The baseline methods for developing increasingly complex models discussed in Section 4.2.1 were also implemented. For each of these baselines, the same process of implementing a simplified model, synthesizing a policy for the model, and then evaluating the transference of this policy was followed for each increasingly complex model. These results are shown in pictorially in Figure 5.3 and summarized in Table 5.2.

Considering Binary Transference first, shown in Figure 5.3a and the second column of Table 5.2, the proposed method shows very favorable results. The naive method, as expected, shows poor results.[2] The heuristic method, which in this case was implemented by considering the magnitude of the phenomena/element, showed significantly improved results over the naive method. Considering the simplicity of this heuristic, these are impres-

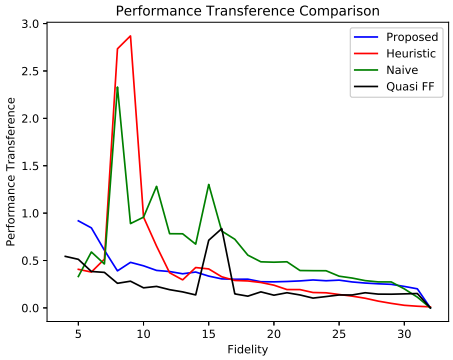
[2]Note, that while the aggregated results across all considered systems did show improved transference compared to the single system in Figure 5.3a, it still acts as the expected lower-bound on transference.

Table 5.2: Summary of measures for transference between proposed method and baseline methods.

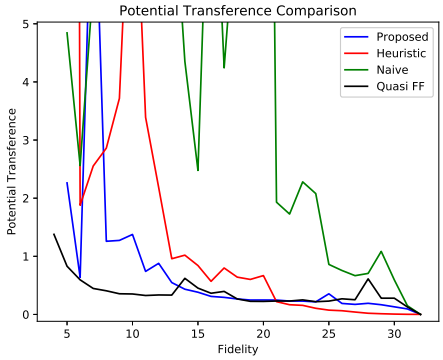
Method	Area Under Transference Curve		
	Binary (Higher is Better)	Performance (Lower is Better)	Potential (Lower is Better)
Proposed	0.74	0.32	0.64
Naive Selection	0.52	0.53	7.99
Heuristic Selection	0.66	0.40	3.24
Quasi-Full-Factorial	0.90	0.22	0.34



(a) Binary Transference



(b) Performance Transference



(c) Potential Transference

Figure 5.3: Transference evaluations for policies synthesized on varying fidelity levels for models designed in order of identified phenomena criticality. The results for the 50 experimental systems have been aggregated here. The left figure shows binary transference rates. The middle figure shows mean performance transference for all successful transfers. The right figure shows mean potential transference for all successful transfers.

sive results. However, not only does it not achieve the same transference as the proposed method, it would be difficult to construct such an accurate heuristic for more complex systems.

Comparing the proposed method to the quasi-full factorial method is a bit more complex. For binary transference, the quasi-full factorial method does outperform the proposed method. It recognizes simpler models that achieve transference, and achieves near uniform transference before the proposed method. There are two things holding this method back though. First, as was brought up in Section 4.2.1 when it was first discussed, it may be infeasible to implement on practical systems. These systems showed transference relatively early, but this method blows up in terms of simplifications required to evaluate if this early transference doesn't happen. Even for these systems that did show early transference, the method required evaluating an average of more than 30,000 simplification systems to achieve this curve. This is more than three times the number of systems that were sampled for the proposed method for a relatively marginal increase in area under the transference curve, and this is for a favorable system.

The second shortcoming of the quasi-full-factorial method was less expected, but possibly more egregious. The method shows non-monotonic tendencies with respect to fidelity and transference. That is, there were cases where adding any additional phenomena to the model actually reduced transference rates. This is something that was expected for the naive method, as there may be distracting phenomena or coupled phenomena that must be added to a model together to get a benefit. It appears that the quasi-full-factorial method as implemented may lead to paths of model development that lead to single-step dead ends. Because there are no false phenomena considered here, those that are modeled but are not actually present in the real system, this is especially surprising. This is another attractive feature of the proposed method, as it was the only method evaluated that showed fully monotonic behavior for Binary Transference.

While Binary Transference was the major focus of this work, it is still worth while to

consider the two quantitative measures of transference. As with Binary Transference, the naive method showed significantly worse results than all other methods considered and acts as a minimally necessary condition for any method to be considered successful.

The comparison for quantitative measures of transference is more complex. On aggregate, the proposed method outperforms the heuristic method for both measures of quantitative transference and is either competitive with or outperforms the quasi-full-factorial method, as seen in Table 5.2. However, the best performing method seems to depend on where in the fidelity space the considered model is, unlike Binary Transference. This is clearly seen in Figure 5.3. That is, at the lowest levels of fidelity, the quasi-full-factorial method provides the best quantitative transference. For high levels of fidelity, the simple heuristic method gave the best quantitative transference. For moderate fidelity levels, the proposed method was either competitive or alone as the best method.

Looking at these graphs, two things are apparent. First, both the proposed and quasi-full-factorial methods appear to hit a plateau in the quantitative metrics once they achieve consistent binary transference. This makes some sense, as binary transference is their main metric for discrimination between phenomena. Once this is no longer distinct, there is less information value to extract and which phenomena to include next is less obvious. The inverse seems to be true for the heuristic method. The magnitude of the phenomena is most likely to affect quantitative measures, but less likely to affect the qualitative evaluation. Once consistent transference is achieved, this takes over and allows for further reduction of the quantitative measures.

While it wasn't explored in this work, it's possible this insight could be used to further augment the proposed method by incorporating quantitative measures of transference in the evaluation of phenomena criticality. Regardless, these results support both Hypothesis 1.0 and Hypothesis 2.0. That is, the proposed method of phenomena criticality evaluation paired with a basic simplification model development strategy results in significantly improved transference for simplified models of a referent system when compared with rea-

sonable baselines and approached the performance of the less feasible quasi-full-factorial method.

5.2.2 Experiment 2: Effects of Sampling Distribution

The previous section discussed the proof of concept experimentation. In general, this showed the proposed method for phenomena criticality measurement to have attractive qualities and that it compared favorably with alternative methods for simplification selection. This and the following two sections will now discuss experiments that try to evaluate the effects of different decisions within the proposed measurement framework. This section is focused on the first of these decisions, the strategy that is used when sampling simplifications to evaluate. This was laid out in Research Question 1.1. Hypothesis 1.1 posited that if the simplified systems are sampled according a representative distribution, the resulting criticality measures will lead to higher levels of transference when compared to criticality measures derived from samples of other distributions.

When discussing this question and developing this hypothesis in Section 3.1.1, it was noted that in much of the literature on designing experiments, such as [92], that representative samples are often the most productive. However, these are not the only possible sampling strategies that should be considered. When developing this experiment in Section 3.3.2, two alternatives to this representative sampling were discussed that were based on sampling from a distribution on the fidelity space before sampling from the simplification space directly. These were discussed to address the fact that a representative sampling strategy will largely sample from simplified models of moderate fidelity. So, the two proposed alternatives would instead enforce a uniform or triangular distribution on the fidelity space.[3]

To evaluate these alternative sampling strategies, each was implemented in a modular

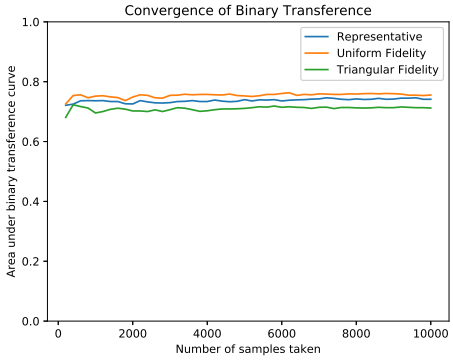
[3] As was discussed when these were proposed as alternatives, these aren't truly uniform or triangular. This is because the number of simplifications at the extreme ends of the fidelity space are severely limited. So, true uniformity can't be achieved without repeating sampled systems. The exact distributions observed can be seen in Section B.2

implementation of the proposed phenomena criticality evaluation framework. The same 50 linear systems as used in evaluating Experiment 1 and outlined in Section 5.1.1 were again used here. Two major considerations were considered in evaluating the differences between the alternative sampling strategies. First, like the previous experiment and as will be used throughout the remainder of these experiments, the area under the transference curves that are achieved when the full method and set of increasingly complex simplified models are evaluated will be used. Second, the effects of sampling density on these results will be considered. this is important to consider, as it is expected that sampling density will affect each of these proposed distributions differently. Also, if two sampling distributions perform similarly for high sampling densities, but one approaches these results for lower sampling densities, it can be used as an additional discriminating factor. As such, the resulting areas under the transference curves are shown across a range of sampling densities in Figure 5.4, with results for the highest sampling density considered are summarized in Table 5.3.

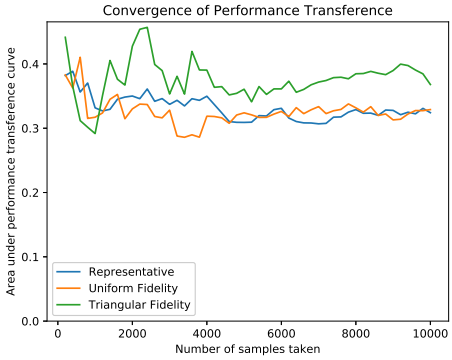
Table 5.3: Summary of measures for transference between proposed method and alternative sampling strategies. Numbers given for the highest number of samples, 10,000.

Sampling Strategy	Area Under Transference Curve		
	Binary (Higher is Better)	Performance (Lower is Better)	Potential (Lower is Better)
Representative	0.74	0.32	0.64
Uniform Distribution of Fidelity	0.76	0.33	0.68
Triangular Distribution of Fidelity	0.71	0.37	0.75

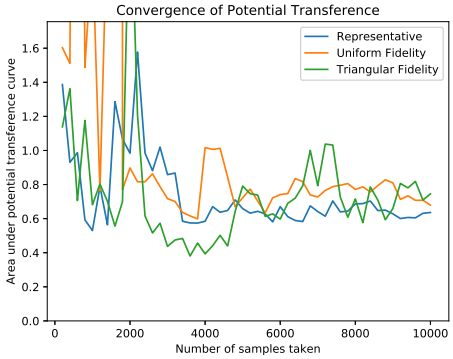
When looking at the summarized results in Table 5.3, it becomes clear that the sampling strategy has a somewhat small but notable effect on the resulting transference curves. The representative sampling strategy gave either the best, or close to the best, transference properties for the three metrics considered. However, it does lead to slightly worse Binary Transference results across the range of sample densities tested when compared with the Uniform Distribution, as seen in Figure 5.4a.



(a) Binary Transference



(b) Performance Transference



(c) Potential Transference

Figure 5.4: Area under the 3 main transference metrics considered for the three alternative sampling strategies for varied sampling densities. The specific curves that lead to these areas can be found in Section B.2.

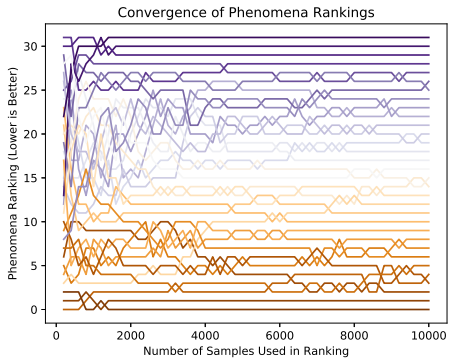
When considering quantitative measures of transference, shown in Figure 5.4b and Figure 5.4c, the picture is less clear. While the proposed representative sampling strategy has the best transference for both Performance and Potential at the highest sample density considered, there isn't a consistent stratification of the different sampling strategies across all considered densities. However, the representative sampling strategy proposed seems to converge to consistent values earlier than the other two methods. That is, it seems to converge within a given bound for each faster, implying that the density considered is sufficient. The other two sampling strategies still show significant variance in Potential Transference, Figure 5.4c, at the highest sampling densities considered, implying that this may not be their true values.

Another way to see this is to consider the convergence of the actual phenomena rankings for each system. This is shown in what are called "Horse Race" plots, shown in Figure 5.5. Figure 5.5a shows one of these plots for a single system. Each point along the x-axis represents a different sampling density. The placement along the y-axis represents the ranking of a phenomena with respect to its criticality, with lower rankings implying higher criticality.[4] There are two things that can be noted from this. First, phenomena at the extreme ends of criticality are identified fairly quickly using the proposed method. Second, there is considerable confusion for the rankings for phenomena with moderate criticality even when a relatively large number of samples were considered. This is partially because the difference between these criticality values is small, so small changes in value can results in large changes in ranking.

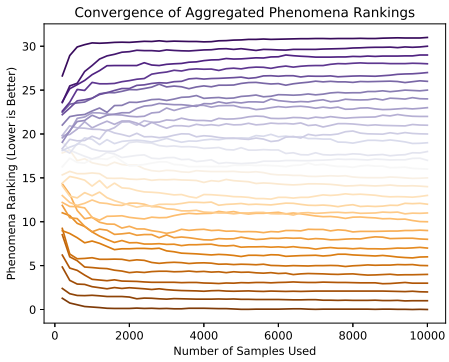
While Figure 5.5a shows the results for a single system, the remaining plots in Figure 5.5 show aggregated results for each sampling strategy. That is, it takes the eventual ranking for a phenomena at the highest sampling density for a system, then tracks where that phenomena was ranked at different sampling densities and averages this across all of the considered systems. This gives a smoother representation of the resulting rankings than would be realistically expected from any single system, but does allow for the general performance for each distribution to be compared. Looking at Figure 5.5b, one thing immediately stands out: these rankings converge much faster than the remaining two distributions. The phenomena at the extreme ranges of criticality have largely converged before even 2000 samples were taken, well before the uniform and triangular distributions of fidelity. Even phenomena with moderate criticalities were mostly sorted out by the time 5000 samples had been taken, again before the two other distributions.

This, combined with the result on the actual transference curves discussed previously, largely supports Hypothesis 1.1. That is, it is reasonable to use the representative sampling

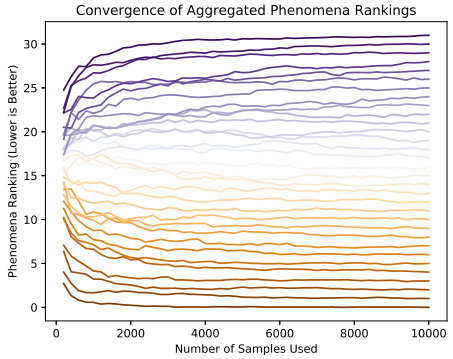
[4]While this figure only shows results for a single system, it is representative of common results seen for many systems and all three sampling strategies. Additional figures for single systems can be found in Section D.2.2.



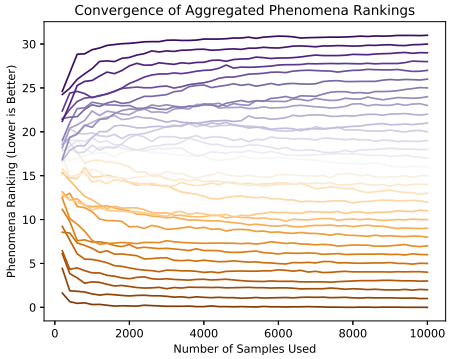
(a) Single System



(b) Aggregated Representative Sampling



(c) Aggregated Uniform Distribution of Fidelity



(d) Aggregated Triangular Distribution of Fidelity

Figure 5.5: Convergence of phenomena rankings for various sampling strategies. The top left shows the results for a single system under consideration that are representative of many of the systems considered. The remaining three plots show these results aggregated for each sampling strategy under consideration. That is, it takes the eventual ranking for a phenomena at the highest sampling density for a system, then tracks where that phenomena was ranked at different sampling densities. So the lowest line represents where the eventual most important phenomena was ranked on average throughout the sampling density range for the different sampling distributions.

strategy originally proposed to determine phenomena criticality. This leads to transference curves as seen in Figure 5.3, and generally outperform the alternative sampling strategies considered.

While the results generally support Hypothesis 1.1 in reference to Research Question 1.1, some additional discussion is required. The expectation underlying Hypothesis 1.1 is that the moderate fidelity models sampled under the representative strategy will reveal the greatest information with regards to phenomena importance within a simplified model.

While that was found to be true for the linear systems considered, this may not be true in general. Given the similar overall performance, it's reasonable to expect that the specific ordering may change for other classes of systems. For example, a system that only exhibits transference for the highest fidelity models would likely benefit from the triangular distribution, as no information can be extracted if none of the sampled models transfer. Similarly, if nearly all models produce transference, information regarding phenomena criticality will be most dense at the low end of the fidelity spectrum. This would favor the uniform or even a reverse triangular distribution. While none of this would lead to a total refutation of Hypothesis 1.1, it does add some necessary nuance to the resulting conclusion.

5.2.3 Experiment 3: Effects of Comparison Metric

Following the proof of concept experimentation discussed in Section 5.2.1, it was shown that the nominal method for evaluating phenomena criticality had merit. Then, experimentation conducted and described in Section 5.2.2 evaluated the effects of different sampling strategies within this method. This section will now discuss a second decision to be made when implementing this phenomena criticality evaluation method. This was laid out in Research Question 1.2 as *which transference measure should be used when evaluating the sampled simplifications?* Hypothesis 1.2 posited that evaluating models by Binary Transference would lead to better measures of criticality, and therefore better transference curves on the resulting set of simplified models, than criticality measures derived from Performance Transference.

As was discussed in Section 3.1.2, both Binary Transference and Performance Transference have different benefits and drawbacks. Binary Transference is a qualitative measure that assesses the likelihood of a policy trained on the simplified system being successfully applied to its goal task on the true system. This simplicity is a strength, as well as its broad applicability. That is, any simplified system can have its Binary Transference evaluated. In contrast, Performance Transference, a measure of the difference in predicted and ac-

tual performance when a policy has been transferred, can only be reasonably measured for systems that show positive Binary Transference. Consider the linear systems as described in Section 5.1.1. Failure in Binary Transference implies an unstable controller. It would be meaningless to evaluate the performance of the policy in this case. As such, Binary Transference can be measured on a wider range of simplifications, possibly giving greater information on which phenomena are critical to transference.

This simplicity comes at a cost in resolution though. As the name implies, Binary Transference can only take a value of either 1 or 0 for a given simplification within the context of evaluation for this method: it either does or does not produce a transferable policy during evaluation. For more sophisticated implementations, it's possible the likelihood of transference could be obtained through repeated evaluation, but this would be a rough estimation that would be likely to yield little additional information and come at significant computational cost. Performance Transference, by contrast, is a continuous variable by definition. This higher resolution information may allow it to overcome the relatively narrow set of simplifications it can be applied to. Considering these two primary comparisons, on breadth of applicable simplifications and resolution of the metric, it is expected that breadth of application will be most relevant and therefore Binary Transference will yield better evaluations of phenomena criticality.

As described in Section 3.3.3, this was evaluated similarly to Experiment 2: Effects of Sampling Distribution above. That is, the same 50 linear systems were again used to define the truth systems for control policies to be trained. Then, the phenomena criticality method was applied twice. First, Binary Transference was used in the third "Evaluate, Normalize, and Score Simplifications" step of the method. The resulting phenomena criticalities were used to define a set of models of increasing fidelity. These models were used to train policies for controlling the linear systems, and their transference curves were identified. Then, the same process was implemented, except Performance Transference was used as the evaluation metric. In cases where Binary Transference did not occur, the sampled model

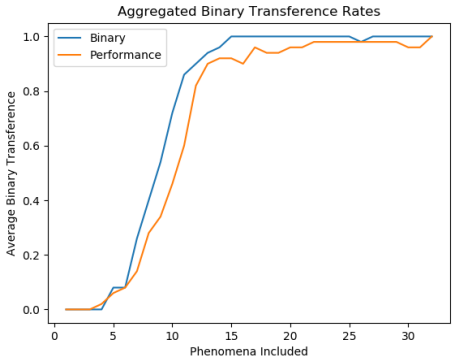
is ignored in the analysis. As previously discussed, this effectively reduces the sample size of the simplifications considered, but increases the information that is gained from each individual sample. This produced a second set of transference curves, for Binary, Performance, and Potential Transference, respectively. These curves are shown in Figure 5.6. These curves can be compared qualitatively for trends and quantitatively by comparing the areas under the curves, summarized in Table 5.4.

Table 5.4: Summary of measures for transference between proposed method and alternative sampling strategies.

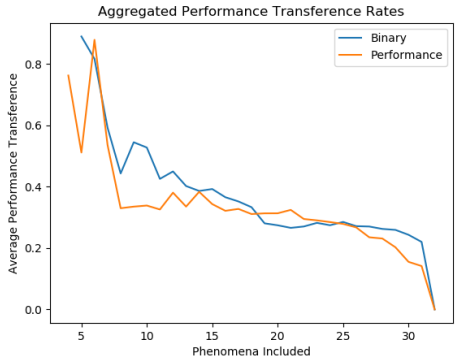
Evaluation Metric	Area Under Transference Curve		
	Binary (Higher is Better)	Performance (Lower is Better)	Potential (Lower is Better)
Binary Transference	0.74	0.32	0.64
Performance Transference	0.69	0.30	0.74

Looking across the three graphs, it seems as though Binary Transference does lead to better evaluations of phenomena criticality. Look at the Binary Transference curve in Figure 5.6a, using Binary leads to greater transference across all fidelity levels when compared to using Performance the evaluation metric. Additionally, using binary transference yields a monotonic increase in transference. This is a desirable property as it implies the identified order doesn't lead to counter intuitive situations where adding additional phenomena to a model actually reduces transference. This is not true of the models built through using Performance Transference, where this counter intuitive situation does occur. This comparison is also confirmed looking at the quantitative measure for area under the curve in Table 5.4, where the Binary evaluation led to an area of 0.74 compared to an area of 0.69 for Performance.

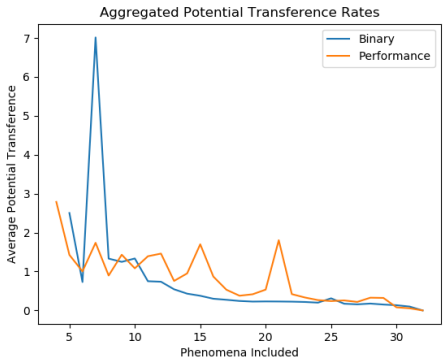
Looking at the quantitative measures of transference yields similar results. It would be reasonable to expect using Performance transference as the low-level evaluation would lead to a better overall Performance Transference rates, but this doesn't seem to be the case.



(a) Binary Transference



(b) Performance Transference



(c) Potential Transference

Figure 5.6: Comparison of transference curves for the method using different metrics during the evaluation of individual sampled simplifications. In general, these curves show that using Binary Transference as the basis for measuring phenomena criticality leads to improved results when compared with using Performance Transference.

Looking at Figure 5.6b, The results are largely similar. While the curve produced from the Performance-based criticality measures is lower overall, with an area of 0.30 vs an area of 0.32 for the Binary-based measures, the differences are not significant. There are even small portions of the fidelity space where the two curves trade domination, implying both were very similar at evaluating criticalities in this respect.

When considering the potential transference curve, shown in Figure 5.6c, Binary Transference appears to be the better evaluation metric. The overall area under the curve for the Binary-based models was 0.64, notably lower than the 0.74 for the Performance-based models. This is also qualitatively true throughout the graph, where the Binary-based curve is nearly uniformly lower. The only area this doesn't hold is for very low fidelities, where

evaluating the data individually revealed a likely outlier, causing the noticeable spike.

Overall, these results all support Hypothesis 1.2, that Binary Transference is a better metric for low-level evaluations within this phenomena evaluation method. This does come with the caveat that this may be system dependent. Due to the nature of these two measures, it's reasonable to expect that for systems where transference is relatively easily achieved will favor the higher resolution of Performance Transference will be a better choice. In general though, it is expected that Binary Transference will be useful for a wide range of systems and is reasonable for a first implementation.

5.2.4 Experiment 4: Effects of Referent Fidelity

Experiments 1 through 3 above outlined the major questions regarding the method to evaluate phenomena criticality. While this looked at important aspects of evaluating the importance of different phenomena to capture within a system model, the results from each made a critical assumption. That is, the referent model used to evaluate transference for the sampled simplifications was the truth model. While useful in evaluating the fundamentals of the method, this is not a good assumption to make for more realistic systems. Part of the motivation for using simulation models is the difficulty or expense associated with using the true system. As such, it is reasonable to question how these results will generalize when the referent system is itself a model of the true system. This was discussed at length in Section 4.1, and was captured by Research Questions 2.1. Hypothesis 2.1 posited that if the referent model used to measure phenomena criticality could not itself be used in training transferable policies, than the phenomena criticality measures would not hold when attempting to develop further simplified models of the true system.

In discussing this question in Section 4.1, it was noted that trying to find a robust measure to positively answer this question may be a fool's errand. One of the reasons transference from simulated environments to the real world is so difficult is because there are countless phenomena that may affect the real system. For complex systems, these phenom-

ena may interact in unexpected ways that make it near impossible to certify whether the simulated model will hold in the real application. However, as is identified in Hypothesis 2.1, it is possible to identify where the referent model will get you into trouble. That is, if the referent model itself can't produce transferable policies, it would be silly to expect the rankings of its associated phenomena to hold any truth when looking at further simplifications.

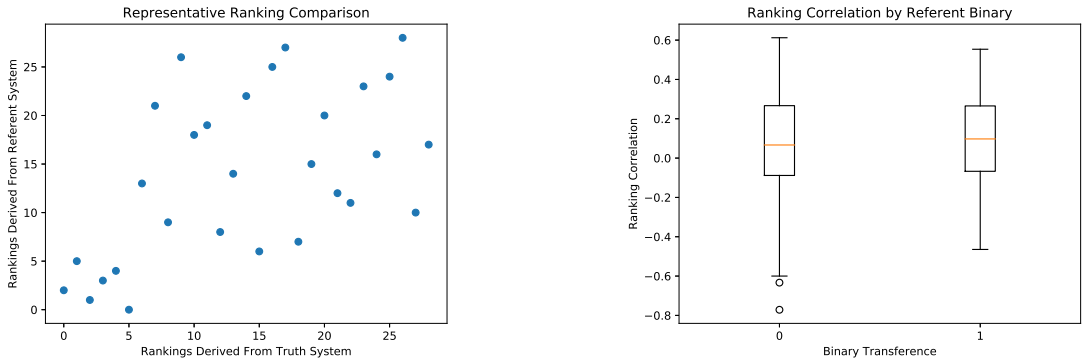
To evaluate the effects of the fidelity of the referent on these rankings, the same linear systems as used in the previous experiments were used. This allowed for "truth" rankings to be considered as the rankings for each system taken from previous experiments. Then, for each system, a set of referent models was sampled from it's possible simplifications. As such, these referents had varied fidelities and transference properties. In this way, the possible effects of many different referents for many different systems could be considered.

For each referent system, the method was applied to rank the phenomena that made it up. That is, if a referent only contained phenomena $(3, 5, 7, 10)$, these four phenomena were ranked relative to each other for their importance with respect to generating policies that would transfer to the referent. Say this gave the following order: $(10, 3, 7, 5)$. This order would then be compared to the equivalent truth rankings. So if the full truth order was something like $(..., 10, ..., 5, ..., 7, ..., 3, ...)$ the compared truth order would be $(10, 5, 7, 3)$.

It was expected that the phenomena rankings based on the method applied to the simplified referents would correlate with the rankings that would be produced evaluating the method with access to the truth model. That is, if we ranked the phenomena captured by the referent by using the proposed method, then found the equivalent rankings using the true system, we would expect these rankings to be at least somewhat correlated. Phenomena that are found to be important to create transference to the referent should also be important for transference to the true system. In this case, it would be expected that the correlations for rankings would be significantly higher for referent models that produced transferable polices when compared to referent models that did not. That is, the relative

rankings of the phenomena should be more similar if the referent model itself is capable of producing transferable policies.

This process is visualized in Figure 5.7. First, we can look at Figure 5.7a to see how we get these correlations. The results shown in this figure are somewhat representative of many of the results that were seen on a referent by referent basis. In this case, the referent system was capable of producing transferable policies. Along the x-axis, the phenomena are sorted according to their rankings when the truth system is used to measure the phenomena criticalities. That is, when evaluating sampled simplifications, they are evaluated based on transference to the true system. Note that this does not consider all possible phenomena for the linear systems, 32 in this case, but only the phenomena present in the referent model. Along the y-axis, the points represent the ranking of the criticality of the phenomena when evaluated with respect to transference to the referent system.



(a) Representative Ranking Comparison

(b) Correlation Distribution Comparison

Figure 5.7: Comparisons between phenomena rankings based on true referent systems and simplified referent systems. To the left, an representative example of the comparison showing weak correlation between the two orderings. To the right, the aggregated results of the orderings separated based on the ability of the referent system to produce transferable policies.

So, if the phenomena were sorted according to their truth rankings, we would get a straight line for Figure 5.7a. Clearly, this is not the case here. While there is a clear delineation between a small cluster of the most important phenomena and other phenomena, the rankings within these groupings seems to be largely uncorrelated. So while for this refer-

ent there would be some correlation between the simplified rankings and the true rankings, it would be weak at best. In general, this was true of many of the referents and systems considered during this experiment.[5] As a general observation, many of the referents that did produce transference had a similar grouping pattern. Phenomena would be grouped similarly, but largely uncorrelated within the group. Others would show good results for many of the phenomena considered, but random phenomena would deviate in ordering significantly. Others were almost entirely uncorrelated.

Figure 5.7b shows these results for each system and associated referents aggregated and separated based on the ability of the referent models to produce transference properties. Clearly, there is little distinction between the two distributions. That is, there doesn't seem to be any relationship between a referent's ability to produce transferable policies and the consistency of it's phenomena rankings with the true rankings. This certainly does not support hypothesis 2.1, that there will be a significant difference between the resulting simplifications based on if the referent model transfers or not.

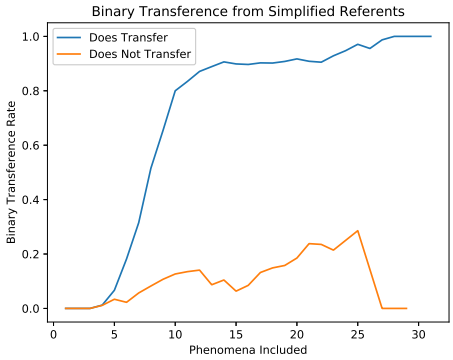
However, we can look at the results of Experiment 2 for an idea of why these correlations did not hold. That is, if we look at Figure 5.5a, it's clear that there is not significant agreement on the specific ordering of phenomena within a model even when looking at large sample sizes on the same system. While the most important and least important phenomena are sorted relatively consistently, the moderate importance phenomena are largely a mess. As such, the correlations between the rankings for the greatest sample density and more moderate sample densities would be weak at best. Even so, by looking at Figure 5.4 that the resulting transference curves are largely similar. That is, while we would expect there to be some correlation between phenomena orders, that may not be the best evaluation criteria. Instead, we should look at the resulting transference curves. These can be seen in Figure 5.8, with summarized area statistics shown in Table 5.5

These figures and the results in the table are much more promising for the method. That

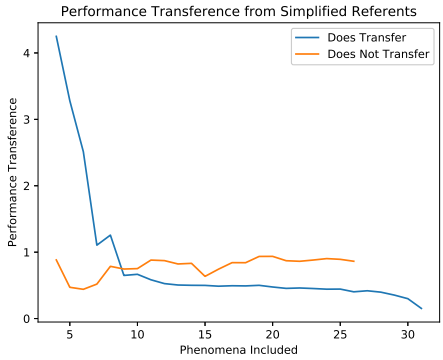
[5] Additional representative examples can be found in Section D.3.2

Table 5.5: Summary of measures for transference of policies trained on simplified models developed through the proposed criticality measures as applied to various referent system classes.

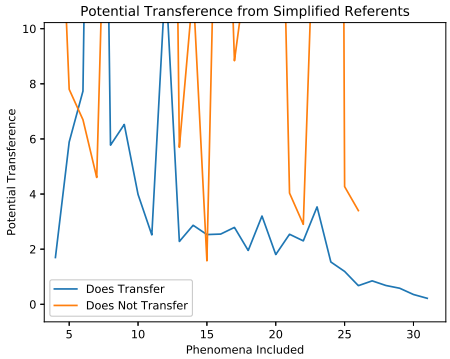
Referent Class	Area Under Transference Curve		
	Binary (Higher is Better)	Performance (Lower is Better)	Potential (Lower is Better)
Truth System	0.74	0.32	0.64
Transferable Simplification	0.69	0.72	3.87
Non-Transferable Simplification	0.10	0.57	18.95



(a) Binary Transference



(b) Performance Transference



(c) Potential Transference

Figure 5.8: Transference curves produced by the proposed method when a simplified referent is used during evaluation.

is, when looking at Figure 5.8, the transference curves produced by applying the method to simplified referents that do produce transferable policies and the curves produced by

applying the method to simplified referents that do not produce transferable policies, there is a clear distinction. This is most clear in looking at Binary Transference in Figure 5.8a. The results for referent systems that do produce transferable policies look largely similar to the results when the referent and truth system were considered one and the same, the results for referent systems that do not produce transferable policies look significantly worse. This is confirmed by considering the area under each curve, shown in Table 5.5.

Looking at the values in Table 5.5 again though, another clear distinction jumps out. That is, the values for transferable referents are similar with respect to binary transference, but far off when considering the two quantitative measures of transference. For performance transference, this is largely explained by the few very low fidelity models that contribute heavily to the area under the performance transference curve. For the potential transference curve, though, there is significant difference. While the transferable referents do produce better results than the non-transferable referents, there is still a significant gap before the potential of the truth systems is matched. This implies that while further simplification from a referent model may still allow for the development of generally successful policies and even accurate predictions of their performance, there is still a significant loss in possible performance when compared with policies trained directly on the truth system.

Taking all of this together, Hypothesis 2.1 is still supported. That is, these results have shown how important having a reasonable referent system for evaluation is to this method. If the referent system used does not produce transferable policies itself, any evaluation of phenomena importance is likely a fool's errand. However, for the cases where a quality referent can be developed and is desired to be simplified further, this method shows significant promise.

5.2.5 Experiment 5: Practical Case Study

The four previous experiments were all meant to evaluate different aspects of the proposed phenomena criticality evaluation method. First, Experiment 1 showed that the method was

reasonable, and produced beneficial results when compared with other baseline methods. Experiment 2 evaluated the effects of sampling strategy on the method, and Experiment 3 evaluated the effects of different evaluation metrics used within the method. Experiment 4 then evaluated how differences between the referent model used for these evaluations and the true target system would affect the usefulness of the resulting phenomena criticality measures. Each of these revealed important aspects of the methodology and provided additional information with regards to its implementation. However, these experiments were evaluated using a relatively simple truth system, a class of 4 state, 4 input linear systems, discussed in Section 5.1.1.

While linear systems continue to see use in much of the literature and practical applications, they themselves are a simplification of the real world. It has been said that "the classification of mathematical problems as linear and nonlinear is like classification of things in the universe as bananas and non-bananas." That is, nearly all known real world systems are nonlinear in nature. It's reasonable then to question whether this method will generalize to more realistic nonlinear systems. To address this, a modified form of the Acrobot system was discussed in Section 5.1.2. This nonlinear system shares many features that commonly face robotic systems, such as nonlinearity, underactuation, and chaotic uncontrolled behavior. So, while still simple in the grand scheme of possible systems to consider, it is an excellent candidate to evaluate the proposed method for evaluating phenomena criticality and the paired simple model development strategy proposed by Hypothesis 1.0 and Hypothesis 2.0, respectively.

To evaluate these with respect to the Acrobot system, we consider the 10 phenomena noted in Section 5.1.2. This gives 1024 possible simplified models that can be used to train a control policy for the system. 100 of these possible models were sampled according to the representative simplification distribution. A policy to swing up the second link in the Acrobot system was trained using an asynchronous version of the DDPG algorithm. [71] Details on this asynchronous augmentation and training parameters used can be found

in Chapter C. The policy was formulated such that it had the same form as that used in [71]. That is, the actor and critic functions were approximated as a neural network with two hidden layers each. Both networks had 400 neurons in the first hidden layer, with 300 neurons in the second hidden layer. Both networks used the *elu* nonlinearity for neuron activation. The final output node of the critic network was a *linear* activation, while the final output node of the actor network was a *tanh* activation.

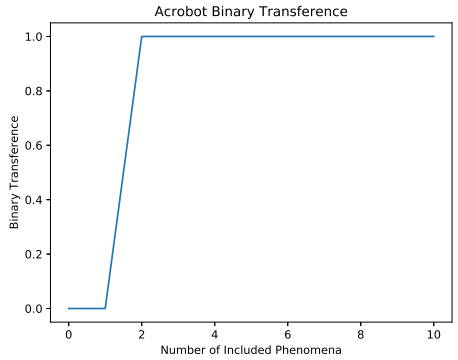
Each policy was allowed to train for 10 hours under the same computational conditions to account for limited resources that are often used for training of these systems. This is in contrast to training for a set number of episodes, which is also commonly done in many reinforcement learning approaches from the literature. This is because it's possible that some of the simplifications would be easier to learn, leading to shorter episodes. This advantage is one of the major motivations for using simplified systems for training in the first place, and using constant numbers of episodes may dampen this benefit. As this work is interested in evaluating the models used for training a policy and not the algorithm itself, it is important to maintain this benefit.

Each policy was then evaluated for transference to the truth system and the remainder of the method followed to evaluate the criticality of each phenomena. The resulting criticality measures are summarized in Table 5.6. These criticalities were used to create a set of increasingly complex models according to Hypothesis 2.0, whose transference properties were evaluated. Curves similar to those from Experiments 1-4 were expected as proof of the method's applicability to more realistic systems. These resulting curves are shown in Figure 5.9. In this case, there are no comparisons to alternative methods. However, we can compare the resulting curves with those for the linear systems from Experiments 1 through 4. While these aren't perfectly analogous comparisons, similar results will further support the claim that this method can be generalized.

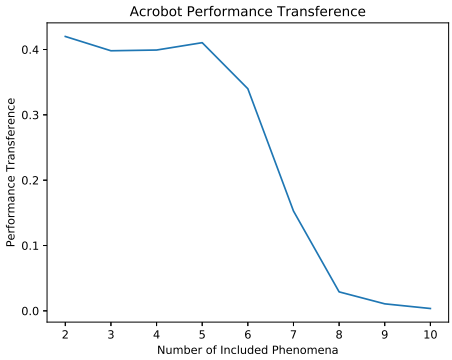
Looking at the curves, these are largely similar to those seen in previous experiments. The binary transference curve achieves transference at fairly low fidelity levels, and main-

Table 5.6: Summary of phenomena criticality scores for the ten phenomena considered for the acrobot system.

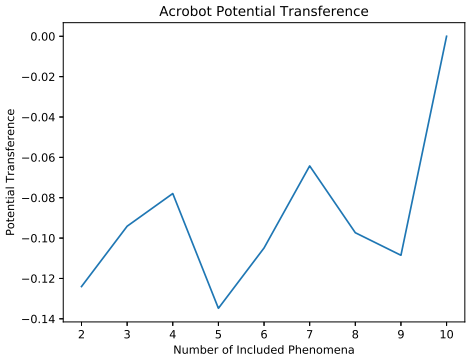
Gravity	Elbow Friction	Elbow Drag	Shoulder Drag	Continuous Torque
0.70	0.10	0.06	-0.03	-0.07
Torque Noise	Elbow Spring	Shoulder Friction	Shoulder Spring	Torque Limits
-0.07	-0.12	-0.21	-0.22	-0.25



(a) Binary Transference



(b) Performance Transference



(c) Potential Transference

Figure 5.9: Curves for transference of control policies for the augmented Acrobot system. These curves show similar levels of transference as those seen in the linear systems experiments, supporting the generality of the method.

tains transference throughout the rest of the fidelity space. In terms of the area under this curve, the Acrobot system had an area of 0.60 below the curve. This is less than that of the method as applied to the linear systems, but is comparable. The quantitative measures are less similar, but show similar trends. For Performance Transference, this is a general

decrease throughout the fidelity space. However, unlike the linear systems case where Performance Transference reached a plateau, there seems to be a fidelity threshold where Performance Transference rapidly improves once crossed.

The Potential Transference Curve is also significantly different. The most notable difference is the negative values for across all fidelities. This is a surprising result given the form of Potential Transference, shown below:

$$T_{Potential} = \frac{\mathbb{E}_{\tau \sim \zeta_R, \pi_R}\left[f\left(\tau\right)\right] - \mathbb{E}_{\tau \sim \zeta_R, \pi_S}\left[f\left(\tau\right)\right]}{\left|\mathbb{E}_{\tau \sim \zeta_R, \pi_R}\left[f\left(\tau\right)\right]\right|} \tag{5.9}$$

This implies that the policies trained on the simplified versions of the system actually outperformed the policy trained directly only the true system. This undercuts one of the key assumptions made when proposing Potential Transference as a metric of interest in Section 3.3.1. That is, that training on the true system directly would produce the optimal policy with sufficient time. While this may still be true, it illuminates a major issue in reinforcement learning: sufficient training time is highly system dependent. Some systems are much more difficult to learn than others. There is even the existence of so-called *wicked systems*, where short-term rewards are actually misleading when long-term returns are considered.

While the acrobot system and reward structure for this problem are unlikely to be truly wicked, the full problem may be complex enough to slow training significantly. So, the use of the same training resources may have led to sub-optimal performance. This provides an excellent example of one of the main motivations for using simplified systems in the first place: by first solving a simplified version of a problem, you can take advantage of the simplicity to learn a reasonable approximation of the behavior before attempting the full problem. By iteratively approaching the true system in the training environment, incremental steps can be taken to account for and take advantage of phenomena independently. This may lead to higher quality policies than training directly on the truth system itself, and is the basis of the field of transfer learning.

One thing to note on this idea of transfer learning is that it may work for small changes in the environment, but it cannot account for qualitative changes in the environment. That can be seen in the lack of transference at the lowest levels of fidelity. Investigating this a bit closer, we can look at the importance of each phenomena and consider its physical meaning and impact on the acrobot system. These scores can be seen above in Table 5.6.

First, it is clear that gravity was the most important phenomena to include in any simplified model. This makes perfect sense, as the omission of gravity would qualitatively change the problem. Any simplified model that does not include gravity will not generate a behavior that adds the energy required to raise the first link as necessary. It's important to note that the spring acting at the first joint could play a similar role if its spring constant was significant enough. However, due to the parameters chosen for the system, the effects of this spring force were an order of magnitude below the effects of gravity.

On the opposite end of the importance spectrum was the inclusion of torque limits in the model. Again, this makes perfect sense and was expected based on the design of the system and policy network. Because the policy network used a tanh activation function for its output node, it already places a limit on the possible torque that can be requested. These limits were intentionally aligned to give this phenomenon minimal impact on the model it could verify at least a portion of the ordering. As expected, this was found to be the least important phenomena to consider.

However, this was not a significant difference from other unimportant phenomena. A possible explanation for this is due to the action space noise applied during training of the policy. That is, when the policy is used for training, it's outputs aren't directly applied to the system as inputs. Instead, some noise added to the torque requested by the policy. In this case, this noise follows the Ornstein-Uhlenbeck process [125], a mean reverting alteration of normal brownian motion. Because of this noisy application, the requested torque may be outside of the bounded range provided by the policy directly. For models with unlimited torque, this allows the training process to explore areas just beyond the regular bounds.

This may have a beneficial effect, allowing the gradient calculation used in DDPG to better represent actions near the limits of the action space. This then gives a further push towards saturation that may be beneficial to the policy overall, that wouldn't be seen in the full model where torque is limited.

This is an example of how using a simplified or other altered model of a system may actually be beneficial in training a policy for the truth system. To investigate this further, the policy from the model that produced the best performance on the true system was allowed to continue training on the full system. The resulting returns are shown in Figure 5.10. On the far left of the figure, returns are shown immediately after the policy is transferred to the true system and learning first begins. It can be seen that not only is this the region of highest returns, but most robust and consistent returns. Relatively quickly, the bounds on the evaluated returns shift lower. For a brief section, the policy even begins to fail to accomplish the task, shown as the spikes towards -1000 in the figure. This is commonly known as *catastrophic forgetting* in the reinforcement learning literature.

This look at further training is commonly called transfer learning in the literature, where a simplified version of a problem is first learned and further tuned on a series of increasingly complex systems. The results from Figure 5.10 and Figure 5.9c point to a possible secondary benefit from following the proposed phenomena criticality evaluation. The series of increasingly complex models defined through Hypothesis 2.0 could be used as a series of increasingly complex training scenarios to accomplish this chain of transfer learning.

Overall, these result support the measure of phenomena criticality outlined in Chapter 3 and embodied in Hypothesis 1.0. Further, they support the use of these resulting phenomena to define a series of models used for training outlined in Chapter 4 and embodied in Hypothesis 2.0. These models are not only useful for training policies in their own right, but may be useful for improving the policy through a series of training exercises through transfer learning.

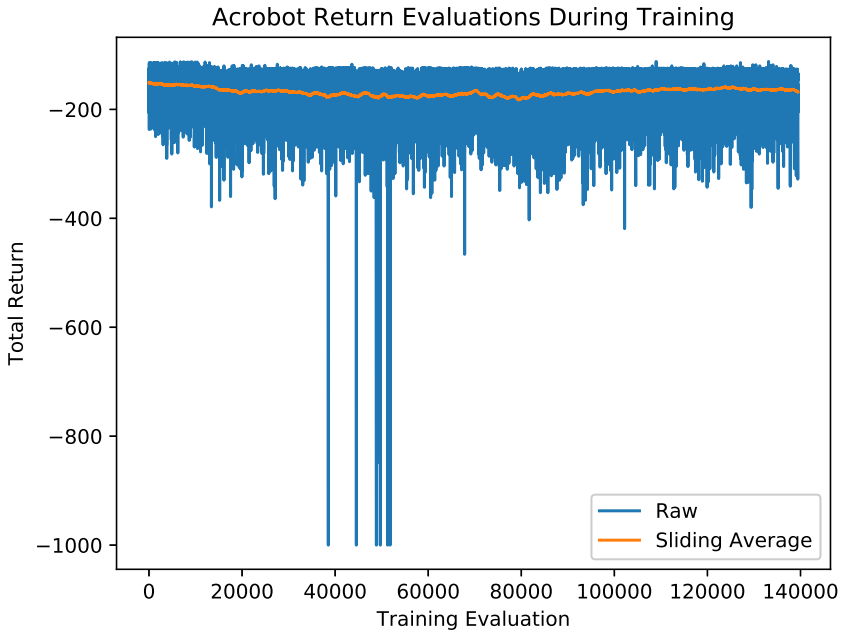


Figure 5.10: Periodically evaluated returns for the acrobot system when continuing training a policy after transference from a simplified system. The sliding average is taken over a 1000-evaluation window centered on the current point.

5.3 Summary

This thesis has set out to study the effects of modeling choices on the transference of policies learning through reinforcement learning in simplified simulations of a system. This chapter discussed the results of the experiments that were designed to evaluate the hypothesis proposed in Chapter 3 and Chapter 4 as part of the research framework for this thesis.

The proposed method of measuring phenomena criticality and associated method of simplified model development were first test against baseline development methods in Experiment 1. As detailed in Section 5.2.1, these results largely supported Hypotheses 1.0 and 2.0. That is, the proposed phenomena criticality measurements lead to simplified models that had significantly improved transference compared to models derived in naive methods or by following a simplified heuristic. Similarly, the proposed method approached the results for the idealized quasi-full-factorial method.

Experiments 2 and 3 then evaluated two major choices within the method: how should the simplifications for evaluation be sampled, and how should these sampled simplifications be evaluated? As discussed in Section 5.2.2, the results largely supported Hypothesis 1.1 that a representative sampling strategy is a reasonable choice. As discussed, it is possible that alternative sampling strategies may be beneficial depending on the behavior of the system. For systems that are difficult to produce transference, a triangular distribution may be more informative. For systems where transference is easier to achieve, a uniform distribution may be more informative. The representative sampling strategy represents a reasonable balance between these two approaches.

Similarly, Section 5.2.3 discussed the results of Experiment 3, which showed that using Binary Transference as the evaluation metric produced better results than using Performance Transference. This was largely attributed to the breadth of systems that can be evaluated by Binary Transference compared to Performance Transference's higher resolution but more narrowly applicable measure.

Experiment 4, discussed in Section 5.2.4, then looked at the affects of using a simplified referent in the evaluation process. This is important for practical systems, as it is rare that transference to the true system would be cheap enough to evaluate as many times as is required by this method. This showed surprising results, in that while the ordering of the phenomena was notably different from the true ordering based on the truth system, the resulting transference curves were largely similar. This also supported Hypothesis 2.1, in showing the importance of having a quality referent for this method to be applicable.

Finally, Section 5.2.5 discussed the results from Experiment 5. This experiment was a re-evaluation of the high level Hypothesis 1.0 and 2.0, but now looking at a more realistic system. This used the Acrobot system, detailed in Section 5.1.2. This is a nonlinear, underactuated system that displays chaotic uncontrolled behavior. These are common problems for autonomous systems, and this system represents a reasonably simple system to develop policies for. These results showed that the method produces similar results on realistic sys-

tems and that the phenomena criticality measures align with expected physical meanings.

Overall, these experiments largely supported the usefulness of this method. Across both the simple linear systems evaluated for Experiments 1 through 4, and the more complex Acrobot system, the method consistently identified simplified models that could be used to train transferable policies.

CHAPTER 6
CONCLUSION

Reinforcement learning represents an attractive approach to developing the next stages of behavior for autonomous systems. While powerful, this is still an emerging field of research. There are significant open gaps that must be addressed before reinforcement learning sees widespread use. Particularly, most frameworks for conducting reinforcement learning require significant exploration of their possible state and action spaces. This is fine for many of the virtual domains that reinforcement learning has been applied to successfully, but will certainly cause issues for systems that operate in the real world.

As such, a significant amount of work has gone into developing reinforcement learning behaviors for real systems in simulations. This allows for many benefits, including enhanced safety, easier recreation of failure cases, and possibly increasing speed of training through parallelization. However, systems trained in simulated environments often fail to transfer to the real world. There are varied reasons for this, but this provided the main motivation for this thesis. As such, the main motivating objective for this work was to identify possible methods for the improvement of models used in simulation-based training of reinforcement learning derived policies.

Chapter 2 discussed relevant background information to further develop this objective. Literature on modern reinforcement learning frameworks was discussed to identify important trends and limitations. This mainly focused on model-free approaches that combined deep learning approaches to policy and value function representations that have seen significant use in recent breakthroughs. Work from the sim-to-real field was also discussed to evaluate current methods for producing autonomous policies that can transfer from the simulated world to the real world. A few major themes were identified. These included attempts to increase the fidelity of simulation models, and domain randomization techniques

that randomized phenomena representations in simulation. Literature from the broader modeling and simulation community was then reviewed to provide addition viewpoints on how modeling and simulation can be done successfully. It was noted that focus on phenomena representation was likely to be useful. As such, the motivation for this thesis was refined to an attempt to develop a method for measuring the importance of different phenomena that have been proposed for a model of an autonomous system. The goal was to develop this measure in such a way that it could be used to identify possible simplified models of a system for training of autonomous behaviors.

Chapter 3 developed this method using a sampling based approach. The goal of the method is to define a useful measure of importance, called *phenomena criticality*, that can be used to identify simplified models of a system to be used in simulations for developing transferable policies. Research questions and associated experiments were proposed to evaluate the method. Chapter 4 then developed a simple model development strategy employing these phenomena criticality measures. As before, the goal was to construct models of a system for use in simulations to develop reinforcement learning based policies that are transferable to the truth system. Additional research questions and experiments were proposed to evaluate the practical considerations of this method.

The work presented in Chapter 3 and Chapter 4 was tightly integrated. Therefore, the results of their proposed experiments were collected and presented in Chapter 5. This covered how the method for measuring phenomena criticality and the resulting simplified models compared with baseline methods, possible alterations to the method, and impacts of imperfect information during evaluation. Overall, the method showed significant promise. When compared with the alternative baselines, the method showed significantly improved transference while approaching a quasi-full factorial search.

6.1 Evaluation of Research Framework

Throughout this work, the motivating objective was to develop a methodology to measure the importance of varied phenomena to be captured in a model for training reinforcement learning based policies in simulation. A sampling based method for measuring this was proposed and pair with a simple development strategy. These were developed in Chapter 3 and Chapter 4 and were framed by the following two research questions:

Research Question 1: *How can the criticality of potential phenomena to be included in a simulation model be compared such that simpler models can produce transferable policies?*

Research Question 2.0: *Given appropriate measures of phenomena criticalities for a referent model, how should simplified models be constructed to achieve the greatest transference with the lowest complexity?*

While in theory, these questions could be answered separately, they are tightly coupled. Any changes to the phenomena criticality measurement would change the resulting model development strategy. Similarly, without models to evaluate comparisons of phenomena criticality lose some meaning. As the main goal was phenomena evaluation, it was proposed that a simple development strategy be implemented. These are captured in the following two hypotheses:

Hypothesis 1: *If a sampling-based approach is implemented to evaluate the possible simplification space, then reasonable simplified models that balance complexity with transference to the true system can be identified.*

Hypothesis 2.0: *If phenomena criticality is measured as proposed and models are developed by including the phenomena in descending order of criticality, then the produced simplified models will show similar or greater levels of transference with fewer phenemena represented.*

These two hypotheses were first tested in Experiment 1: Proof of Concept. This compared the proposed method for phenomena criticality measurement and the simple development strategy with alternative baseline methods. These baselines included a naive method meant to establish a minimum necessary condition for success, a simplified heuristic method that determined phenomena ordering without any evaluation, and an ideal case where simplifications were evaluated in a quasi-full-factorial manner. The proposed method favorably compared with both the naive and heuristic-based methods while approaching the ideal method with significantly few system evaluations.

Given this proof of concept, the method itself was then characterized through a series of questions and hypotheses. First, the sampling strategy used to pick simplifications for initial evaluation as part of the measurement were evaluated. It was hypothesized that a sampling strategy that maintained a similar distribution as the full simplification space would lead to the best results. This was captured with the following research question and hypothesis:

Research Question 1.1: *How should the possible simplifications of a given referent model be sampled for evaluation of phenomena criticality?*

Hypothesis 1.1: *If the simplified systems are sampled according to a distribution matching that of the possible simplification space, the resulting phenomena criticality measures can be used to develop lower complexity models with similar levels of transference.*

This hypothesis was tested in Experiment 2: Effects of Sampling Distribution. The results of this experiment led to a nuanced understanding between the difficulty of the

problem and the choice in sampling strategy. The representative sampling strategy worked well across many cases, and is a reasonable choice as a starting point. However, it is possible that alternative sampling strategies may allow for a lower number of samples to lead to similar results. That is, if a system already shows high likelihood of transference across many simplifications, sampling more densely in lower fidelity areas of the space may be most reasonable. Alternatively, if transference is exceedingly rare strategies that sample from high fidelity portions of the simplification space may be useful. In this way, an adaptive strategy that adjusts the sampling distribution as individual simplifications are evaluated may be worthwhile to investigate.

The next portion of the methodology to be considered was the actual evaluation metric used on each sampled simplification. It was expected that Binary Transference would yield the most generalizable results, as it was the most broadly applicable metric. This is captured in the following research question and hypothesis:

Research Question 1.2: *Given a referent and associated simplified model of a system, how should its ability to produce transferable policies be evaluated?*

Hypothesis 1.2: *If the simplified models are evaluated with respect to their Binary Transference rates, then the resulting comparisons of phenomena criticality will allow for phenomena to be rank ordered in a way that allows simpler models to produce greater levels of transference.*

This hypothesis was tested in Experiment 3: Effects of Comparison Metric. These results were more straightforward, Binary Transference was the superior metric in almost every way. This included both looking at aggregated results and at a system by system level. Surprisingly, Binary Transference was the better comparison metric even for the quantitative comparisons.

For these evaluations, it was assumed there was access to the truth system. However,

this runs counter to much of the motivation for using simulations and simplified models in the first place. To employ this method on a practical system, evaluations of sampled simplifications would have to consider transference to a referent system that is itself a simplification of the true system. It is reasonable to question whether the results will hold in this case, and is captured by the following research question and hypothesis:

Research Question 2.1: *How sensitive are these phenomena criticality measures and the resulting model development strategy to the fidelity of the referent model?*

Hypothesis 2.1: *If the referent model itself does not show reasonable transference to the true system, then the relative criticality measures evaluated with respect to it will not hold when evaluated on the true system.*

Experiment 4: Effects of Referent Fidelity evaluated this hypothesis. In this case, reasonable transference was defined as achieving Binary Transference to the true system. It was expected that their would be a clear separation based on referent system transference in correlation for phenomena rankings when compared to those derived from the true system. The results of initial experimentation on this, however, did not match these expectations at all. Even for referent systems that did show transference, there was little correlation in phenomena rankings. However, when these rankings were used to build models the resulting transference curves showed a clear distinction between referents that did produce transferable policies and those that did not. In general, these transference curves were notably worse than those derived with access to the truth system, but still showed that this method could be applied in practical cases.

Throughout these experiments, the method had been applied to linear truth systems. While linear systems are useful throughout many fields, the world is nonlinear. So, it was important to ensure that the method generalized well to a nonlinear system. To do so, it was applied to a modified version of the Acrobot system [108] in Experiment 5: Practical

Case Study. The results were largely similar, showing that the method is applicable to a range of distinct systems. When evaluating the returned phenomena criticality measures, they matched up well with expectations based on the physical meanings of the phenomena involved. Gravity, a phenomena that fundamentally changes the required approach to the problem, was found to be the most important by far. Torque limits on the actuator were correctly identified as having little effect, largely due to the purposeful design of the policy structure. Additionally, the simplifications identified through this method actually led to improved policies when compared to those that were developed directly on the truth system. This case shows something that has been noted in the broader transfer learning literature, but was encouraging to find.

6.2 Contributions

When considering the development of models for training reinforcement learning based policies in simulation, the existing literature gave little thought to what should be represented. Nearly all systems that even considered the models being used in the training followed a *more is better* approach and just added additional phenomena to a model until transference was achieved. This ad-hoc approach often led to uneven results.

The major contribution of this thesis is a more reasoned approach to model development. This approach is centered on a method to evaluate the relative importance of phenomena, here called phenomena criticality, to be represented in a simplified model of a system. While the work shown here was specifically focused on evaluating the importance in the context of training reinforcement learning based policies, it is possible that it can be extended to other contexts. It was shown that these measurements not only identified obvious orderings, but could handle more subtle comparisons as well. This led to improved transference for simplifications even when compared with heuristics that require full knowledge of a system.

When this measure of phenomena criticality evaluation is paired with a simple model

development strategy, it was shown that simplified models of a system can be produced that maintain similar transference properties as the full models they were derived from. This allows for intelligent simplifications to be made to complex systems. As was shown in the final experiment, discussed in Section 5.2.5, this can actually lead to an improvement in the training of control policies when compared with the full referent model. This could be due to many possibilities, but in general these simplified systems represented easier problems in terms of policy complexity. That is, simpler policies could solve the simplified systems than were required for the full referent. As such, training algorithms could spend more time optimizing the found policies. While initially surprising, this lines up with some previous examples from the broader transfer learning literature. This suggests that simplifications identified through this method can represent a step forward in terms of training capability.

Given the requirements for the application of this method, there are a few potential places in the design cycle where this method can be applied. The first, and primary target considered for this method, is in the development of new behaviors for a preexisting system. That is, some platform has already been designed and fielded for some given task but is now being applied to a new scenario. For an example of this, consider an autonomous helicopter platform similar to the GTMax platform discussed in [56]. This is preexisting platform that has been used in a variety of autonomous missions. Imagine there is interest in using the same platform for short-range package delivery. Reinforcement learning has been proposed as a way to train an end-to-end control algorithm for landing at unprepared sites, including precision site identification within a predefined landing zone.

The existing simulation environment has seen use in developing controller behavior for the system. However, training of this end-to-end behavior will require the simulation to be augmented to account for new sensor modalities, such as vision-based landing site selection. Phenomena to consider would include camera distortions, possible damage to sensors in the field, ground effects both aerodynamically and on the environment for visual sensing, and many others. Given a total list of these possible phenomena of interest, the method

could be applied to ensure the new simulation model is developed to the appropriate complexity that the newly derived landing behavior will have high probability of success when tested on the actual platform. This would involve developing initial models of these new phenomena, evaluating them according to the method discussed in Chapter 3, and determining which are necessary to include such that training can occur. Ideally this process would identify things such as environmental dynamics as non-critical, as these could very expensive to model well.

While exploration of new behaviors for preexisting systems is the primary target, the method is also likely to be useful when applied throughout the design process of a novel system. Take the above example of the GTMax platform. Say there is interest in developing a new platform based off of the lessons learned in testing. The goal is to develop a smaller platform more specifically targeted towards reinforcement learning research. This would likely lead to a desire for a significantly cheaper platform that can evaluate riskier behaviors. The earliest likely application of the proposed method for simulation design of this new system would be near the transition from conceptual to preliminary design. That is, the broad architectural decisions have been frozen, and there is now a desire to beginning finalizing lower level design decisions. At this stage, the configuration has largely been decided upon and previous experience with these configurations can guide the development of a list of phenomena that may impact system performance. Similarly, initial models of a system should already be implemented, with the next phase targeting further refinements of these models. The goal of this method would be to aid in this model refinement such that overly complex models can be avoided. This is especially important for experimental platforms that will face many iterations of behavior design. In this way, behavioral studies can be brought forward in the design cycle such that the system can be optimized in a more holistic fashion.

Applying the method in such a way may require slight alterations. Of these, the method will likely need to be applied iteratively. As the behaviors desired from the system change,

the most relevant phenomena to capture in simulation models are also likely to change. Simplistic landing behaviors, such as constant thrust descent, may give way to more complex landing behaviors. In that example, the method may identify a need for improved ground effect modeling during the next design phase.

To further understand this application, consider a spiral development process that is used in the design of many complex systems. At the completion of each spiral, lessons that have been learned in designing a simplified version of the system are brought forward to inform the next, more complete, design iteration. This methodology could be applied at each transition point between spiral phases to ensure design decisions are being made using the most relevant model of the system. This would allow for changes in behavioral requirements to influence the design of new models of the system.

Other contributions of this work include the introduction of a new metric for evaluation of simulation based training: Potential Transference. This new metric shifted the focus towards the ability of a simulation to train a useful policy. While it's application may be limited when considering complex systems that cannot be trained in the real world, it is a useful reminder that accuracy in behavior prediction may not be the only relevant measure to consider when evaluating a simulation environment.

A smaller contribution, though arguably just as important, is bringing a focus onto considering phenomena representation in models for reinforcement learning. Much of the literature on developing simulation models for autonomous systems takes a *more is better* approach to developing system models which may not be appropriate. This work argues that it is important to consider which phenomena specifically are contributing to the development of useful policies. By evaluating this, with or without the proposed method, other methods for training behaviors can be made more effective.

Finally, a contribution not significantly discussed in the main body of the thesis, but detailed in Chapter C, is a novel implementation of the DDPG learning algorithm, [71], using asynchronous experience evaluation. This was necessary to develop for its speedup

of the base algorithm, but was also found to have a stabilizing affect on many of the results.

6.3 Future Research Directions

When considering the development of models for training reinforcement learning based policies in simulation, the modeling process is often thought of as secondary. This thesis represents a significant step in evaluating the impacts of different phenomena captured in a simulation model on the transference of reinforcement learning based policies. The results discussed in Chapter 5 were promising and showed the proposed method can be used for developing simplified models of a system. However, there are always additional avenues to explore. First among these research directions are applications to more complex systems. While the Acrobot system described in this dissertation presents many of the challenges that current autonomous systems face, it is still a fairly simple system. For more complex systems, such as an autonomous helicopter, there may be additional challenges to overcome.

Specifically, this method assumes that a model of the system that does lead to transference already exists and has been identified. This is especially important for application to practical systems. This comes from Hypothesis 2.1. While the results of Experiment 4, discussed in Section 5.2.4, showed this can lead to reasonable assessments of phenomena criticality, getting this initial referent itself is no simple feat. This has lead to significant research in the area of sim-to-real, as discussed in Section 2.2. While there are methods for establishing these referents, this is an important limitation to consider, as the results of Experiment 4 are a double-edged sword. If the model used as a referent for this method does not itself produce transference, then much of the outputs for this method are meaningless. As was discussed in Section 2.2, a recent approach to developing transferable behaviors is using domain randomization during simulation-based training. [120] To address the shortcomings of non-transferable referent systems, it would be interesting to see if wrapping this method in a domain randomization framework would be helpful. It's possible that this

randomization of the referent would allow for similar results in terms of transference of these criticality measures.

On its own, domain randomization seems to be a powerful tool in producing transferable policies but has significant limitations. Current applications are very intensive, relying on ad-hoc approaches to determine what to randomize. Some very recent approaches, such as [15], [74] and [82], have taken an automated approach to developing these randomization procedures. However, these often rely on a black box approach that doesn't incorporate domain knowledge of a problem, or are limited to defining distributions for a preset list of parameters. It would be interesting to take the criticality measures defined by this method as a basis for determining the parameters to perturb within a domain randomization framework. This could be a powerful way to combine improved model development strategies with training frameworks that produce robust policies. This could be implemented in an iterative approach that adjusts randomization profiles based on the current policy as well.

Focusing more narrowly on the method itself, further research into adapting these ideas of phenomena criticality could be applied in a predictive framework could be useful. That is, instead of using the measures produced by the method simply to compare phenomena for inclusion in simplified models, the measures could be adapted to signify a prediction of a simplified model's transference properties. The first step would be to investigate how these measures could be transformed to produce a prediction of Binary Transference. Simple transformations, such as the softmax transformation commonly used in classification problems for machine learning, could be applied and evaluated. This would represent a first step, with follow on research to evaluate a similar idea for quantitative measures of transference.

In a similar manner, the results shown throughout this work used direct knowledge of the referent system for developing the simplified models. The phenomena representations were taken directly from the truth model. This may not be possible for practical systems, and inaccuracies in representation may have a significant impact on the results. It is ex-

pected that the results will hold for approximated representations of a phenomena based on system identification, but this has not been evaluated extensively. This is something to keep in mind when using these evaluations on more complex systems where the true representation is unknown and deserves further study.

When simplifications were first discussed in Section 2.3, it was noted that their are three rough classifications of simplifications: omission, aggregation, and substitution. While the argument was made that all of these simplification classes can be represented by omission alone, these representations may not be the most natural or meaningful for a problem. For example, consider parameters that are evaluated on some scale where resolution is a major consideration, like the timestep used in a simulation. While this can be represented as omission, with each phenomena now represented by an array of evaluations along time and different timesteps omitting different instances along these arrays, this is not a natural representation. Clearly, this approach will lead to dimensional explosion in the evaluation. As such, it would worthwhile to explore how this method can be applied to other types of simplifications that are commonly required. Similar applications to systems that do not have cleanly separable phenomena, such as neural network representations of systems, could also be fruitful.

Smaller technical details could also be explored to further enhance the method. These include other normalization schemes during the scoring phase, automated sampling distribution adjustments, and tighter integration with statistical tests for differentiating phenomena. While it is expected that these will have small impacts on the overall process, they may be useful in specific cases. Similarly, it may also be fruitful to pursue integrating this method into model-based reinforcement learning strategies. This work was focused on model-free learning algorithms, but it may be applicable to model-based strategies as well. These strategies have many attractive features, such as improved sample efficiency, more physically intuitive structures, and potential for guarantees on performance. These come at the expense of developing and integrating a model of system behavior. Many of the issues

from the broader modeling literature are relevant in developing these internal models, and so this method may find use here as well.

Finally, while this method primarily focused on developing estimates of phenomena criticality for modeling systems to be used in training reinforcement learning in particular, it may have broader use. Current autonomous systems face large open questions in terms of their certification for safety critical systems. Aerospace systems face particular challenges in this regard, as common approaches to failure mitigation like "stop and reset" cannot be applied. [18] As was mentioned in Chapter 1, there is current work developing hierarchical control structures for unmanned aircraft systems that would allow for a management system to switch to specialized controllers to recover from misbehaving autonomous control. These were considered out of scope in developing the methods proposed in this work, but may represent a potential use case.

One example of this is the run-time assurance architecture defined in ASTM F3269-17. [109] In developing this standard, it was recognized that verification of commonly used autonomous behavioral algorithms are likely to be too complex to verify with classical means. Examples of these complex algorithms include the deep neural network based policies developed as part of Experiment 5 for this work. As such, a higher-level architecture is required that evaluates incoming and internal data streams to ensure the platform remains in a safe operating envelope for these behaviors.

For this and similar methods to be applicable though, there needs to be a solid understanding of the safe operating envelope for these complex behaviors. The only way to do this with reasonable risk profiles during initial development of the behavior is through simulation. This is specifically called out, with the possible inclusion of flight test results, in requirement 5.1.5.2 parts (2) and (3) of the ASTM standard. [109] However, as has been discussed throughout this work, the transference of behavior from simulation to the real world for these complex functions is hard to guarantee. So, it stands to reason that the safe operating envelope derived from simulations of these behaviors would also fail to transfer

unless a proper simulation is designed.

This is very similar to the framing used in this work for identifying the phenomena to capture in simulations used to train reinforcement learning behaviors in simulation. For this use case, instead of using phenomena to be modeled, possible state thresholds on the safe operating envelope could be used. Similarly, instead of evaluating transference of behaviors to a referent model, failure cases could be used. In this way, the method outlined in this work could be used as a starting point to define a sampling-based approach to identifying safe operating envelopes for the use of complex control algorithms.

6.4 Summary

As autonomous systems continue to grow in capability, the need for policy development strategies that can handle more abstract goals has become apparent. Reinforcement learning has shown itself capable in many domains, and is an attractive field to further develop. This is still a relatively young field, and there are many opportunities to expand. One of the most critical areas to address for reinforcement learning to be useful for autonomous systems acting in the real world is simulation-based training. Within this, addressing the *reality gap* between simulated and real worlds is critical.

It is clear from the literature that at least a portion of this gap is due to effects of modeling choices when developing simulation environments. This thesis focused on improving our understanding of these effects. This was accomplished through a novel method for evaluating the importance of representing different phenomena withing these models. The developed technique was shown to produce encouraging results with respect to transference from simplified models to truth systems. It was also shown to identify simplified models that can be used to train better performing policies than those that were derived directly on the truth system. This represents an important step on the path towards wider usage of reinforcement learning based systems.

Appendices

APPENDIX A
EXPERIMENTAL SYSTEM DEFINITIONS

This section will provide detailed descriptions of the systems considered for possible reproduction. This is in addition to the descriptions provided in Section 5.1. First, additional details on the linear systems and their simulation are described. Then, additional details on the Acrobot system will be further detailed.

A.1 Linear Systems Experimentation

Experiments 1 through 4 used linear systems as the truth system. These linear systems experiments were of the classic form below:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{A.1}$$

For these experiments, these were 4 state, 4 input systems. So, we can expand this equation as:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{A.2}$$

Each element for the A and B matrices was drawn from independent uniform distributions over the range $[-1, 1]$. 50 such systems were generated to produce aggregated results for each experiment to ensure the conclusions drawn were not one-off coincidences for a particular systems. Simulation of these systems used a simple forward-Euler time stepping scheme with a constant Δt of 0.01 seconds.

Expanding the linear system in this way, it is clear that if we treat each element as an individual phenomena there are 32 total phenomena for this system. Following from the method as described in Chapter 3, we can then describe this and simplifications of this system using 32 bit strings with each bit representing the inclusion or omission of an individual phenomena. While the ordering of the phenomena at this stage is arbitrary, for consistency a simple procedure was implemented. Both the A and B matrices were reduced to row vectors by concatenating all rows in order. Then, these two row matrices were concatenated to create a single list of phenomena for the model. The full referent would be represented by a string of 32 1s. For an example simplification, consider the string 01001010000110110011110000011100. This would represent the following system:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & a_{12} & 0 & 0 \\ a_{21} & 0 & a_{23} & 0 \\ 0 & 0 & 0 & a_{34} \\ a_{41} & 0 & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & b_{13} & b_{14} \\ b_{21} & b_{22} & 0 & 0 \\ 0 & 0 & 0 & b_{34} \\ b_{41} & b_{42} & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{A.3}$$

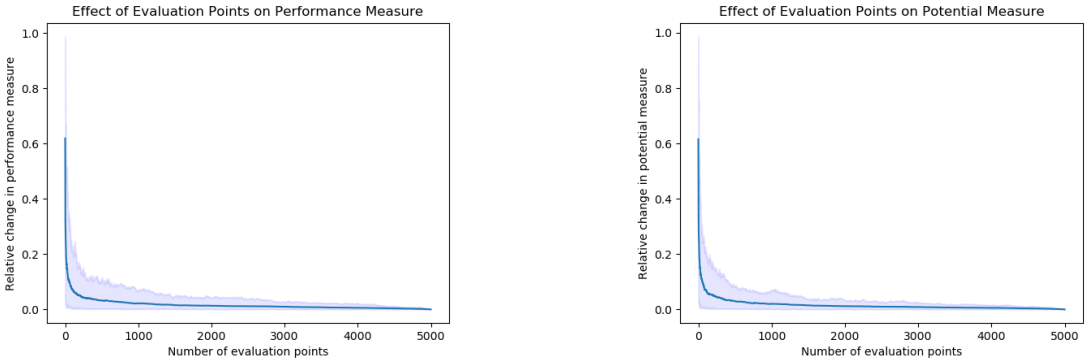
To evaluate the performance of a controller for these systems, points were randomly sampled from the appropriate unit-ball and simulated until the system had approximately converged to the equilibrium point at the origin, or 10,000 time steps had occurred.

In order to get a measure of the performance of a given controller on a given linear system, a linear quadratic cost function along the trajectory was used. This was formulated as shown below. Here, Q and R both represent weight matrices of appropriate dimensions. For this case, these were set to identity.

$$C = \int (\mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}) dt$$

It is important to note that this cost measure is highly dependent on the initial point of the trajectory. This dependence stems from both the magnitude of the initial point and the location of the point with respect to the system matrix eigenvectors. The impact of the

magnitude was addressed by sampling initial points from a unit ball centered on the origin. To ensure the dependence on location did not bias the resulting transference metrics, a number of initial points were randomly sampled from the surface of a unit-ball centered on the origin and the resulting costs averaged to calculate the quantitative transference metrics. The number of samples to take was determined by a simple study to evaluate when the when these metrics converged. The results of this study is shown in Figure A.1.



(a) Performance Transference metric convergence study.

(b) Potential Transference metric convergence study

Figure A.1: Results of a convergence study for the number of sample points used to evaluate the quantitative transference metrics.

This showed that after sampling 500 initial states from the unit ball, the average measures of transference had largely converged to within 5% of their values when 5000 initial states were used. Further sampling showed diminishing returns as this was well past the knee in the rolling average curve. As such, each transference measure used 500 randomly sampled initial states in its evaluation for comparisons of quantitative transference.

As only linear policies were considered for the linear systems experiments. These policies were taken directly from an analytical derivation of the LQR controller with unit identity cost matrices. As was discussed in Section 5.1.1, this is a shortcut to approximating reinforcement learning based policies. For well trained reinforcement learning based policies, they should closely approximate the optimal policy. In the case of linear systems under linear quadratic cost, this optimum policy can be derived analytically in the form of

the LQR controller. As such, using the LQR controller in place of training a new policy from scratch allows for a significant speedup in the evaluation process for the proposed method with little expected impact on the resulting conclusions. Given the results of Experiment 5 for nonlinear systems and policies trained from scratch using an altered form of DDPG, this assertion seems reasonable.

A.2 Acrobot System

The Acrobot system was largely detailed in Section 5.1.2. That is, its behavior is governed by the following system of equations:

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = 0 \tag{A.4}$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = \tau \tag{A.5}$$

Where

$$\begin{aligned}
d_{11} &= m_1 l_{1,c}^2 + m_2\left(l_1^2 + l_{2,c}^2 + 2l_1 l_{2,c}\cos q_2\right) + I_1 + I_2 \\
d_{22} &= m_2 l_{2,c}^2 + I_2 \\
d_{12} &= m_2\left(l_{2,c}^2 + l_1 l_{2,c}\cos q_2\right) + I_2 \\
d_{21} &= d_{12} \\
h_1 &= c_{1,1}\dot{q}_1 + c_{1,2}\dot{q}_1\left|\dot{q}_1\right| - m_2 l_1 l_{2,c}\sin q_2 \dot{q}^2 - 2m_2 l_1 l_{2,c}\sin q_2 \dot{q}_1 \dot{q}_2 \\
h_2 &= c_{2,1}\dot{q}_2 + c_{2,2}\dot{q}_2\left|\dot{q}_2\right| + m_2 l_1 l_{2,c}\sin q_2 \dot{q}_1^2 \\
\phi_1 &= k_1 q_1 + \left(m_1 l_{1,c} + m_2 l_1\right) g\sin(q_1) + m_2 l_{2,c} g\sin(q_1 + q_2) \\
\phi_2 &= k_2 q_2 + m_2 l_{2,c} g\sin(q_1 + q_2)
\end{aligned}$$

For these equations, $c_{i,j}$ represents the damping coefficient for joint i of the j^{th} order

damping, k_i represents the spring constant for the spring located at the i^{th} joint, and $l_{i,c}$ represents the center of mass for the i^{th} link. The implemented system used the following values for each parameter:

Table A.1: Settings for individual Acrobot parameters during experimentation

Parameter	Value
m_1	1
m_2	1
l_1	1
l_2	1
$l_{1,c}$	0.5
$l_{2,c}$	0.5
I_1	1
I_2	1
k_1	0.1
k_2	0.1
$c_{1,1}$	0.1
$c_{1,2}$	0.01
$c_{2,1}$	0.1
$c_{2,2}$	0.01
g	9.81

As was noted in Section 5.1.2, this system then had 10 phenomena under consideration. These were torsional springs located at the fixed joint and at the joint between the two links of the pendulum, linear and quadratic damping terms at the fixed joint, linear and quadratic damping terms at the joint between the two links of the pendulum, inclusion of gravitational forces, continuous application of torque as control input, limitations placed on control torque available, and whether or not the torque is considered directly as requested or noisy.

The first seven of these have clear links to individual parameters in the governing equations shown above. The last three however are specific to the application of torque, τ. To understand their effects, we must discuss the policy implemented for the system. The policy took a similar structure of that given in the original DDPG paper, [71], as this was shown to be a reasonable starting point for many systems and produced good results for

the modified Acrobot. That is, the policy was defined by an Artificial Neural Network with two hidden layers. The first layer consisted of 400 nodes, with the second hidden layer consisting of 300 nodes. Each node used the Exponential Linear Unit. This is similar to the common Rectified Linear Unit used in the original paper and through much of the machine learning literature, but has a smooth derivative function. This was shown to have a positive effect on training stability during initial development. The final output node of the policy used a $\tanh$ activation function to bound output between -1 and 1. This matched the bounds for the system when torque was considered limited. This was chosen to minimize the effects of this phenomena, allowing for a positive identification of phenomena ordering and a useful sanity check if the method gave a different least important phenomenon.

Given this, the torque applied to the system can then be represented as:

$$\tau = \min\left(\tau_{max}, \max\left(\tau_{min}, \pi_\theta\left(s\right)\right)\right) + \omega \tag{A.6}$$

$$\omega \sim \mathcal{N}\left(0,\ 0.1\right)$$

That is, if torque limits are applied to the control input, they are applied before the noise is applied. And if noise is applied to the system, it is applied in an additive sense and drawn from a Gaussian distribution with zero mean and variance of 0.1. For simplifications that instead use a discrete version of torque, the torque is mapped to the nearest available setting. These settings are defined as 11 equally spaced settings on the interval [-1, 1].

Given all of this, the dynamics are then propagated using a basic implementation of the 4th order Runge-Kutta integration scheme, as implemented in the base OpenAI gym implementation of the Acrobot system. [11] Commands to the system are updated at 5 Hz, with commands held constant between evaluations.

APPENDIX B

DATA GENERATION

B.1 Experimental Framework

Given the primary focus of this research on developing relations between simulation model fidelity and transference of training for reinforcement learning based policies, a generic research framework was developed for producing transference data for the control of dynamical systems. An overview of this framework is shown in Figure B.1. This consists of three major components: model creation, controller synthesis, and trajectory generation. These three components are then implemented on three separate paths: straight through the referent model, through simplified models, and synthesis on the simplified model before evaluating on the referent model.

The referent model is somewhat self-explanatory. It contains information about, and data produced by, the referent system. For most practical sim-to-real problems, this would be ”the real world.” The details for this model are scenario specific. By synthesizing a policy and evaluating this on the referent system, a sense of the potential performance can be evaluated.

The simplified models are then sampled from the possible set of simplifications for the referent model. Each sampled model will omit various phenomena from the referent according to the chosen sampling distribution, as detailed in Section 3.1.1 These models use the same parameter settings as the referent model for simplicity. This is analogous to tuning individual phenomena separately.

After the different simplifications have been sampled, the next step in the experimental process is to synthesize controllers for each. For clarity, this step produces a policy synthesized directly on the referent model, and a set of simplified policies, one synthesized

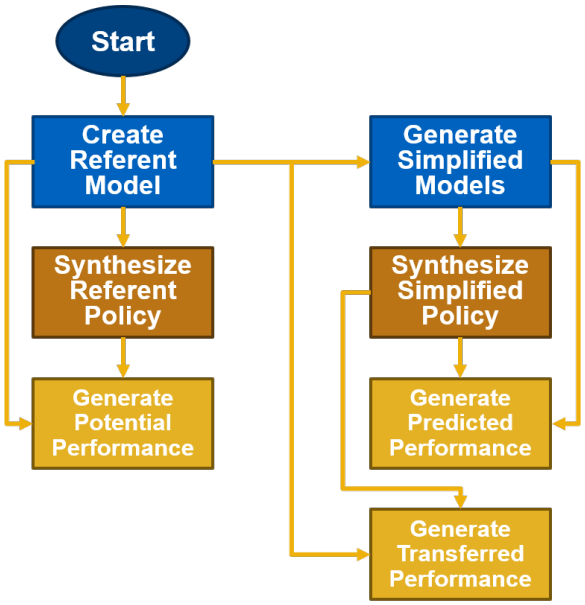


Figure B.1: The generic framework used to conduct experiments for this work. In general, a referent model is created, with controllers synthesized and evaluated on this model. This provides a measure of potential performance. Then simplified models are sampled from the possible set of simplifications of the referent model. Similarly, controllers are synthesized and evaluated for these models. For the simplified models, synthesized controllers are evaluated both on the simplified model themselves, to give a sense of predicted performance, and on the truth model, to give a sense of transferred performance.

for each of the sampled models. The referent policy is meant to act as a baseline when comparing the achieved potential of training in the different simplified models. This measure of potential achievement for policies trained in the simulated environment is useful for evaluating the training effectiveness of the different simplified simulations. The set of simplified policies are the main focus of this work. These policies are synthesized on simplified versions of the referent model as an analog to training in simulation.

The final step in data generation is to evaluate the synthesized policies. The referent policy is evaluated only on the referent model. The simplified policies are evaluated both in their simplified training environments and the targeted referent environment to get a sense of their transference. In this way, Binary Transference is measured simply be evaluating the success of the transferred performance in Figure B.1. Performance Transference is a straightforward calculation evaluating the relative change in the predicted performance

and transferred performance in Figure B.1. It is important to note that this change may not always be negative, as it is possible for a transferred algorithm to surpass its predicted performance due to beneficial phenomena that were not modeled in the simplified environment. As such, this comparison will be made on an absolute change basis, as it would be difficult to predict whether the affects of simplified phenomena will be positive or negative in a general setting.

The second measure of transference will then compare the transferred performance and potential performance from Figure B.1. The reason for this comparison is that a controller trained directly on the referent should achieve true optimal performance for that given policy architecture. When comparing algorithms, such as during the design process for a new autonomous system, we may care more about the potential performance of an algorithm given the ability to continue training and tuning in the real world. As such, we would like to have a simulator that trains a policy such that it is close to that of the true optimal.

This generic framework is used throughout the experimentation discussed in Chapter 5. This allowed for rapid iterations and isolation of issues. For an implementation on a realistic system, only the simplified path would be investigated, as described in Chapter 3.

B.2 Model Simplification Generation

This work mainly considers simplifications of a simulation model through omission. While this may appear limiting, it is argued that the other types of simplification, namely aggregation and substitution, can be viewed as special cases of omission. Additionally, it allows for some very attractive possibilities for the implementation of generating these models. These possibilities are discussed below.

To test varying simplifications, a rapid method for generating models with different inclusion/omission profiles must be developed. For this, a straightforward binary encoding of a simulations phenomena was implemented. This is similar to approaches commonly taken for parameter encoding with genetic optimization algorithms. The basic concept is

to treat each phenomena separately. It's inclusion or omission from a model can then be represented by a single bit of a binary string. That is a '1' implies inclusion with a '0' implying omission. We can then represent any simplified model as a string of these ones and zeros.

This simple formulation for specifying a simplified model has many benefits. First, it allows for a rapid iteration through all possible simplifications for a given system. This can be achieved by simply iterating through all numbers from 0 (the trivial model) to $2^n - 1$ (with n representing the number of phenomena considered) and identifying the binary string for each number. For systems with a small number of phenomena, this allows for a full-factorial design of experiments to evaluate the effects of possible phenomena to include in a model.

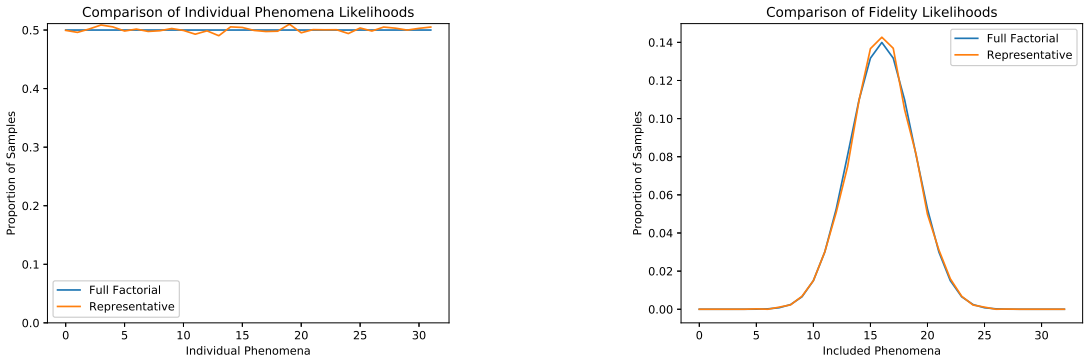
While full factorial designs are attractive to ensure interacting phenomena are considered appropriately, they quickly become intractable for larger systems. Consider a linear system of the form $\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$. Treating every element of the A and B matrices as an individual phenomena, the number of phenomena for the entire system scales with the equation $2^{n*(n+m)}$, where n is the number of states for the system and m is the number of inputs to the system. The number of simplifications for a full factorial design of a linear systems then scales not only exponentially, but quadratically exponentially with the number of states of the system.

This quickly becomes intractable or even small systems: a system with 4 states and 4 control inputs, as used for this experimentation and described in Section A.1 produces over 4 billion simplified systems to simulate.[1] Clearly, the number of systems to consider quickly becomes untenable.

Luckily, using binary strings to represent a given simplified model is the gift that keeps giving. Randomized sampling of these simplifications can be accomplished simply by uni-

[1]To really bring this point home, consider the space necessary to store these string representations. Assuming 32-bit strings, just the string representations of these models would require 17GB of storage for this "simple" system! Let alone any space needed for the systems themselves or the data they produce.

formly sampling across the integers from 0 to $2^n - 1$. This simple approach produces a sampling strategy that maintains representation of the total simplification set. Each phenomena is included in simplifications with likelihood equal to that in the full factorial sampling. Similarly, the likelihood of the total number phenomena considered for any given model (ranging from 0 to n) matches with that produced by a full factorial design. These features can be seen in Figure B.2.

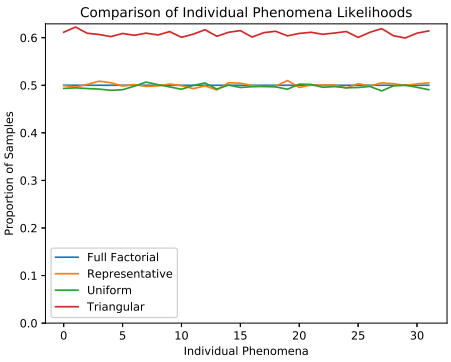


(a) Individual Model Inclusion

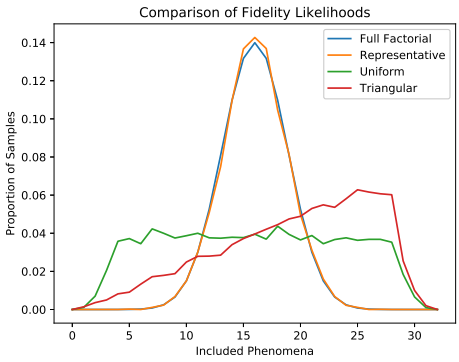
(b) Distribution of Model Fidelity

Figure B.2: A comparison of a full factorial design for modeling simplifications and a representative sampling design for modeling simplifications. Each design yields similar distributions at both the individual phenomena inclusion level and the total phenomena per model level. These proportions are given for a 4 state, 4 input linear system (32 total phenomena) with 10,000 modeling samples taken for the representative sampling strategy.

As can be noted from Figure B.2, this method maintains a similar representation for both individual phenomena and fidelity level. While these two characteristics are the major concerns for a general system, individual system classes may have additional characteristics of interest. For example, systems that are especially tricky to produce transference on may benefit from sampling from higher fidelity simplifications more often. This was discussed in Section 3.1.1 and evaluated in Section 5.2.2. These proposed two alternative methods of sampling based on first sampling a desired fidelity. First was a distribution that maintained a uniform fidelity distribution. Second was a distribution that heavily favored high fidelity models. These alternative distributions are compared with the full factorial and representative distributions in Figure B.3



(a) Individual Phenomena Inclusion



(b) Distribution of Model Fidelity

Figure B.3: A comparison of a full factorial design for modeling simplifications and each alternative distribution for modeling simplifications. The representative and uniform fidelity designs yields similar distributions for the individual phenomena inclusion level, while the triangular fidelity design includes all phenomena more often at an individual level. The representative design is similar with respect to the total phenomena include per model, while the uniform and triangular fidelity designs are as expected. These proportions are given for a 4 state, 4 input linear system (32 total phenomena) with 10,000 modeling samples taken for each alternative sampling strategy.

As can be seen in Figure B.3, these are not perfectly uniform or triangular with respect to sampled model fidelity. That is largely because of a lack of possible simplifications at the extreme ends of the distribution. That is, all possible models follow a binomial distribution with respect to fidelity, so the number of simplified models that includes m phenomena for a referent with n total phenomena is $\binom{n}{m}$. As such, for large numbers of samples, it is impossible to maintain the perfect proposed distributions based on fidelity level.

APPENDIX C
REINFORCEMENT LEARNING IMPLEMENTATION

For policy synthesis, this work used an altered version of the DDPG algorithm defined by Lillicrap et al. [71] The major alteration to this algorithm was the use of asynchronous agents for rolling out trajectories used for training. This is similar to the A3C algorithm, however it follows a direct policy gradient update scheme laid out in the original DDPG paper. That is, there is only a single learning agent in this algorithm, compared with the distributed learning agents in the A3C algorithm. A sketch of the pseudocode for this approach is shown below in Algorithm 1.

The main alteration is the use of asynchronous trajectory generation processes. These processes all begin using the same initial parameters for the policy function. However, they are updated every time they return a completed trajectory to the main learning process. As this is a stochastic process, each agent uses a slightly different parameterization of the actor policy. This results in additional exploration of the space, somewhat akin to parameter space noise discussed in [89], that would not occur for a sequentially learning agent. This was found to have a stabilizing effect on training, given an appropriate number of trajectory processes were used.

When training this algorithm, there are some important hyperparameter relations to consider. Obviously, network architecture choices will have a significant impact on the performance of the eventual policy. The critic architecture must allow for enough complexity to properly capture the reward function used by the training environment. Similarly, the actor architecture must have enough complexity to capture any nonlinearities needed for solution and modes that may be needed for behavioral switching.

Three hyperparameter choices inherited from the original DDPG formulation have an important role in this asynchronous implementation that must be considered further. These

are the number of experiences to maintain in the replay buffer, the number of experiences used to constitute a single trajectory (given the system does not return early), and the number of experiences to sample from the replay buffer at each training step. The interaction between these three choices impact the success of training significantly, as they determine the likelihood of any given experience being used during a training step. A reasonable estimate of this relation can be given by treating each training step as a Bernoulli trial. That is, given the probability of a specific experience being included in any one sample is:

$$p_s = 1 - \prod_{i=0}^{n_s-1} \left(1 - \frac{1}{n_b - i}\right)$$

Where n_b is the number of experiences contained in the replay buffer, and n_s is the number of samples used to estimate the policy gradient at each step. So, the total probability an experience is used for calculating the policy gradient is:

$$P = \sum_{i=1}^{N} \binom{N}{i} p_s^i (1 - p_s)^{N-i}$$

Where N is the number of training steps taken while the experience is in the buffer. Assuming constant length trajectories are added to the buffer at once, as is the nominal case for the asynchronous implementation used, then N can be estimated as:

$$N = \begin{cases} \left\lceil \frac{n_b}{n_t k_s} \right\rceil, & \text{if optimistic} \\ \left\lfloor \frac{n_b}{n_t k_s} \right\rfloor, & \text{if pessimistic} \end{cases}$$

Where n_t is the number of experiences in each trajectory, and k_s is a multiplier for the number of training steps taking before receiving the next trajectory.

The interplay between these three parameters is very important, especially for stochastic environments. If the positive conditions are rare, it becomes more important that they are not skipped by chance. If reward functions are less sparse, then these parameters can be

tuned for performance, as individual experiences becomes less valuable. Increasing the number of background processes for trajectory production does improve exploration of the state-action space for the system, but reduces the number of training steps that are taken before new trajectories are added to the buffer. As such, it is important to consider how many background processes to use, as this does not have a uniformly positive impact on training performance. Possible improvements, such as importance based sampling, could be used to further enhance this method and mitigate downsides of this choice.

For the results shown in Section 5.2.5, both network architectures were taken from [71]. That is, each was a two layer network with 400 nodes in the first hidden layer, 300 nodes in the second hidden layer. The output node of the actor network was a single node with a tanh activation, while the output node of the critic network was a single node with a linear activation. For these results, 6 trajectory generation processes were used. For parameter optimization, the Adaptive Momentum optimizer, or ADAM [62], was used with a learning rates of 0.0001 and 0.001 for the actor and critic parameters, respectively. Policy and critic gradients were estimated using 64 sampled experiences per time step. Each trajectory used a maximum of 1000 time steps (representing 200 seconds). If the success conditions were met before reaching 1000 time steps, the trajectory returned early. Both actor and critic used target network update rates, α of 0.001.

As suggested in the original DDPG paper, [71], exploration was accomplished using an Ornstein-Uhlenbeck process for additive action space noise. This process was parameterized by θ of 0.2 and σ of 0.15. Similarly, weights connected from the final hidden layer to the output nodes were initialized to random values sampled from the range [-1e-3, 1e-3] for both the actor and critic networks to ensure initial actions and expected returns were near 0.

Algorithm 1: Asynchronous Deep Deterministic Policy Gradients

Result: $\pi(s|\theta_\pi)$, θ_π
Initialize replay buffer, R
Initialize θ_π
Initialize θ_Q
$\theta'_\pi \leftarrow \theta_\pi$
$\theta'_Q \leftarrow \theta_Q$
for $n_{processes}$ **do**
  Begin trajectory process in background, assign PID
end
while *True* **do**
  if *Received trajectory,* τ *from background process* **then**
    $R \leftarrow \tau$
    Send updated θ_π to process
  end
  Sample minibatch of N experiences, (s_i, a_i, r_i, s_{i+1}) from R
  Set $y_i = r_i + \gamma Q' \left(s_{i+1}, \pi' \left(s_{i+1}|\theta'_\pi\right)|\theta'_Q\right)$
  Update critic by minimizing loss:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q\left(s_i, a_i|\theta_Q\right)\right)^2$$

  Calculate sampled policy gradient and update policy:

$$\nabla_{\theta_\pi} J \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q\left(s, a|\theta_Q\right)|_{s=s_i, a=\pi(s_i|\theta_\pi)} \nabla_{\theta_\pi}\pi(s|\theta_\pi)|_{s=s_i}$$

  Update target network parameters:

$$\theta'_\pi \leftarrow \alpha_Q \theta_Q + (1-\alpha_Q)\theta'_Q$$

$$\theta'_\pi \leftarrow \alpha_\pi \theta_\pi + (1-\alpha_\pi)\theta'_\pi$$

  if *Stopping Condition* **then**
    Break
  end
end

Algorithm 2: Background Trajectory Generation Process

Result: Trajectory, τ
Initialize θ_π from main training process
while *True* **do**
    Initialize noise process $\mathcal{N}$
    Initialize environment with s_0
    Initialize empty trajectory, $\tau \leftarrow []$
    for *i=1; i ¡ Max Steps; i++* **do**
        $\omega_i \leftarrow \mathcal{N}$
        $a_i \leftarrow \pi(s_i|\theta_\pi) + \omega_i$ Update environment with action a_i and observe new state s_{i+1} and reward r_i
        Append experience (s_i, a_i, r_r, s_{i+1}) to trajectory τ
        if *termination condition* **then**
            Break
        end
    end
    Send trajectory, τ, to main learning process
    Wait to receive updated θ_π from main learning process
end

APPENDIX D
ADDITIONAL RESULTS

Chapter 5 discussed the key results of this thesis. However, there were additional results for many of the experiments that were not directly related to the overall thesis or were excessive for the main body of the work. This appendix will collect some of these additional results.

D.1 Experiment 1: Proof of Concept

D.1.1 Individual Systems

Section 5.2.1 discussed the major results for the proof of concept experimentation. Much of the conclusions were drawn by looking at aggregated transference graphs to give a better sense of the results for the class of linear systems. These results are based on the collection and average of many individual systems. The results for these individual systems follow.

For each individual system, a nominal implementation of the method discussed in Section 3.2 was applied to derive criticality values for each element of the dynamical systems matrices. These were used to build decreasingly sparse representations of the true system matrices. Binary Transference was evaluated with respect to the stability of the controllers derived from the simplified systems, leading to truly binary results. Performance and Potential Transference were each evaluated with respect to standard linear quadratic cost with identity weighting matrices discussed in Section 5.1.1.

Each individual system shows the actual dynamical system considered, and transference curves for the three main metrics considered: Binary, Performance, and Potential Transference. The Binary Transference curves do not show a rate as the policy derived is either successful or not, leading to truly binary results. Ideally, these would show a single transition, and this is the case for the proposed method for the majority of the systems

investigated. However, there was occasional non-monotonic behavior. This was rare.

In aggregate, the proposed method outperformed both the naive and heuristic based methods. For rare individual systems though, either the naive or heuristic method may have been superior. In most cases, the greedy (quasi-full-factorial) method produced the best transference curves. It was occasionally matched by the proposed method, at significantly reduced computational costs.

Figure D.1: System 0

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.1249 & 0.7088 & 0.0948 & -0.7667 \\ -0.5581 & -0.4456 & -0.3132 & 0.5904 \\ 0.4839 & 0.6542 & 0.7444 & 0.4497 \\ -0.4869 & 0.3942 & 0.1233 & 0.5982 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.8876 & -0.4388 & 0.3610 & -0.2844 \\ -0.9776 & 0.5071 & 0.9127 & 0.6731 \\ -0.9363 & -0.7857 & -0.8601 & -0.9804 \\ 0.9671 & 0.6246 & 0.8811 & 0.0692 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

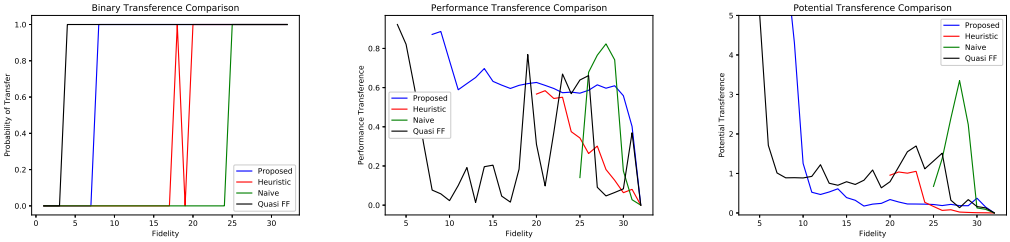


Figure D.2: System 1

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.4148 & 0.4068 & 0.9975 & -0.2455 \\ -0.1024 & 0.6509 & -0.1878 & -0.6526 \\ -0.1220 & 0.2562 & 0.4900 & -0.4740 \\ 0.4428 & -0.3059 & 0.8833 & 0.3272 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.2509 & -0.8633 & 0.4868 & 0.6596 \\ -0.6102 & -0.6867 & 0.1958 & 0.3145 \\ -0.1749 & 0.6693 & 0.4450 & -0.2935 \\ -0.8668 & 0.8447 & -0.6597 & -0.2025 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

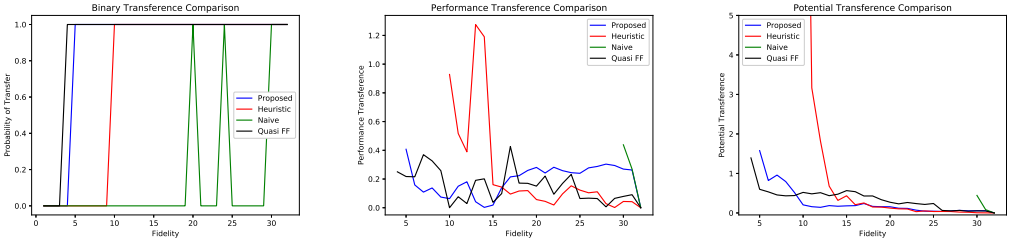


Figure D.3: System 2

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.4413 & 0.4702 & -0.4387 & -0.4167 \\ -0.4360 & -0.7000 & 0.4382 & 0.6401 \\ -0.8016 & -0.6151 & 0.5638 & -0.4396 \\ -0.9705 & -0.7381 & 0.6209 & 0.4400 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.2541 & -0.8558 & 0.9150 & -0.2904 \\ -0.9694 & 0.9749 & 0.2416 & -0.8606 \\ 0.3632 & 0.7037 & -0.7754 & 0.5556 \\ -0.5998 & -0.3124 & 0.2386 & 0.2426 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

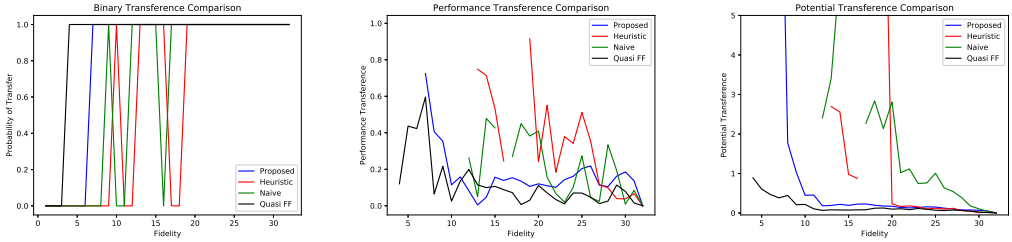


Figure D.4: System 3

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.6458 & 0.2784 & -0.1367 & -0.9431 \\ -0.5882 & 0.8692 & -0.0663 & -0.3214 \\ 0.9730 & -0.2824 & -0.6971 & -0.4374 \\ 0.5823 & -0.6599 & 0.9271 & 0.2923 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.4142 & 0.9041 & -0.6207 & 0.2551 \\ -0.0828 & -0.8526 & -0.7802 & -0.1698 \\ 0.6390 & -0.9816 & -0.5849 & 0.7589 \\ 0.4830 & 0.0002 & -0.0695 & 0.7635 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

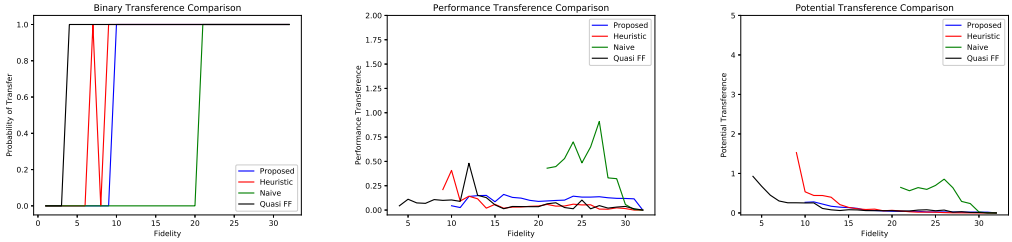


Figure D.5: System 4

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.7460 & 0.8171 & 0.9619 & -0.1128 \\ 0.8247 & -0.3371 & 0.4294 & 0.1837 \\ 0.5195 & -0.7771 & -0.4447 & -0.9346 \\ 0.5873 & -0.3592 & 0.8690 & 0.6846 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.7549 & 0.0572 & 0.5009 & 0.2557 \\ -0.9226 & -0.7534 & 0.1271 & 0.6869 \\ -0.4025 & 0.7857 & 0.9295 & -0.2136 \\ -0.3862 & -0.8532 & 0.3122 & -0.0864 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

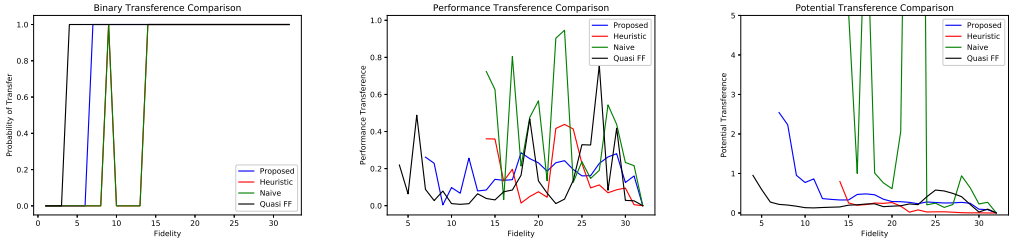


Figure D.6: System 5

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.8062 & -0.7579 & -0.9187 & -0.2513 \\ 0.8847 & 0.7650 & 0.0619 & 0.4219 \\ -0.1698 & 0.9764 & -0.2054 & -0.5871 \\ -0.3143 & -0.4438 & -0.0387 & -0.5854 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.0978 & 0.1902 & 0.3530 & -0.9623 \\ 0.7166 & 0.8257 & 0.9467 & 0.4459 \\ 0.5560 & 0.0006 & 0.8997 & -0.0214 \\ 0.0714 & -0.9948 & 0.5217 & -0.7838 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

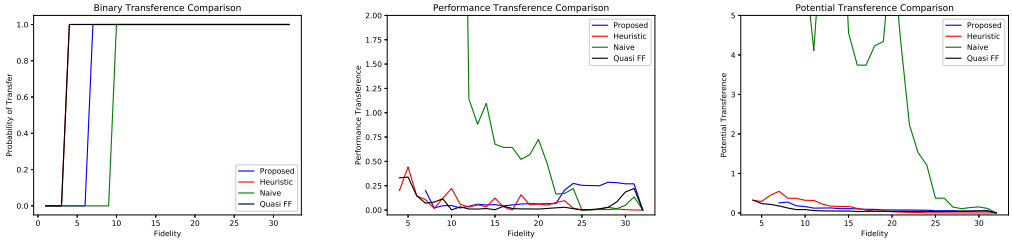


Figure D.7: System 6

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.5751 & -0.6227 & 0.0525 & -0.2659 \\ 0.3840 & -0.6543 & 0.9714 & -0.0087 \\ 0.9403 & 0.5458 & -0.3230 & 0.5489 \\ 0.1679 & -0.1127 & -0.1350 & -0.6681 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.7993 & 0.7064 & 0.4346 & -0.0822 \\ 0.5689 & 0.3676 & -0.4341 & 0.5500 \\ -0.5509 & 0.6497 & 0.2771 & 0.2241 \\ -0.5473 & -0.8229 & 0.2834 & -0.3461 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

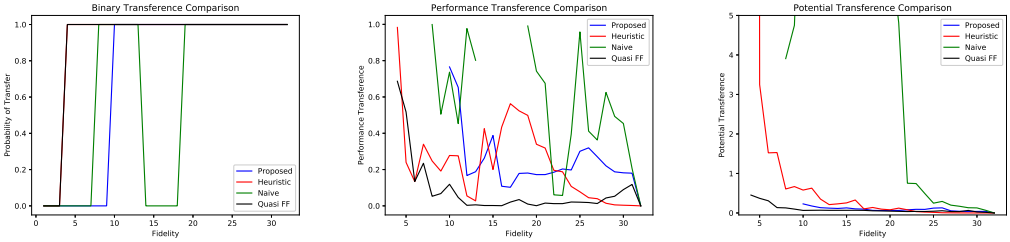


Figure D.8: System 7

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.6413 & 0.5411 & -0.4200 & -0.3727 \\ 0.1941 & 0.4288 & 0.7478 & 0.3734 \\ -0.9206 & -0.0042 & -0.8752 & -0.0713 \\ -0.6346 & -0.0314 & 0.9760 & -0.3036 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.3126 & 0.9300 & -0.6495 & 0.8112 \\ -0.1600 & 0.8475 & -0.7844 & 0.8994 \\ 0.2664 & -0.0278 & 0.3481 & 0.5343 \\ 0.3673 & -0.6782 & 0.0334 & 0.0359 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

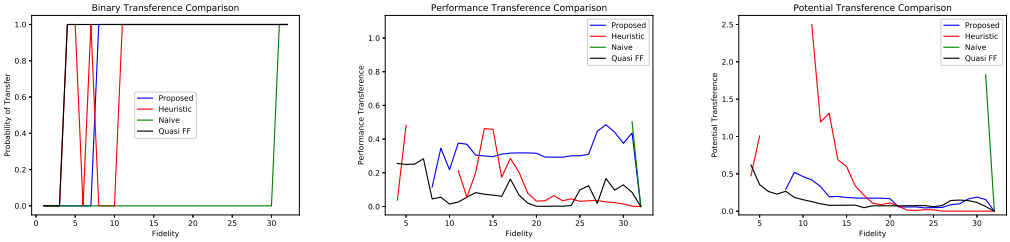


Figure D.9: System 8

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.7258 & -0.0990 & 0.9579 & -0.5301 \\ -0.7658 & -0.8697 & 0.8717 & 0.0769 \\ 0.7787 & 0.8236 & -0.1115 & 0.4899 \\ -0.1050 & 0.4606 & 0.1807 & -0.7169 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.6878 & -0.2132 & -0.9496 & 0.7817 \\ -0.5921 & 0.3867 & 0.3208 & 0.2412 \\ -0.7982 & 0.5498 & 0.0994 & 0.7744 \\ -0.2761 & -0.8881 & 0.5901 & 0.8054 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

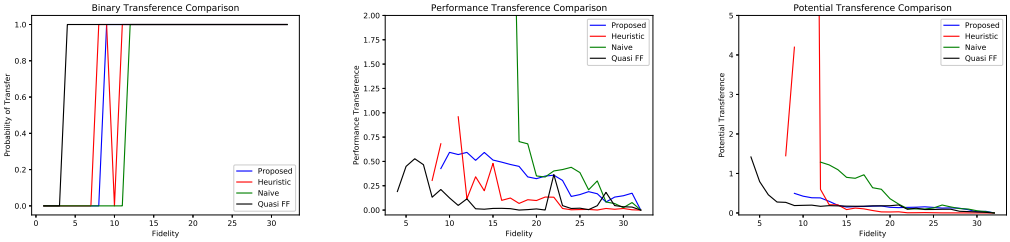


Figure D.10: System 9

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.9906 & 0.8477 & -0.4443 & -0.3583 \\ -0.2540 & -0.3116 & -0.3581 & 0.3471 \\ -0.9489 & -0.1167 & -0.4987 & -0.3158 \\ -0.7990 & 0.3680 & 0.5356 & 0.8415 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.3210 & -0.1029 & 0.6887 & 0.5282 \\ 0.6815 & -0.6316 & 0.1700 & 0.1362 \\ -0.9669 & -0.3625 & -0.2411 & 0.9779 \\ 0.9518 & -0.5011 & -0.1738 & -0.3155 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

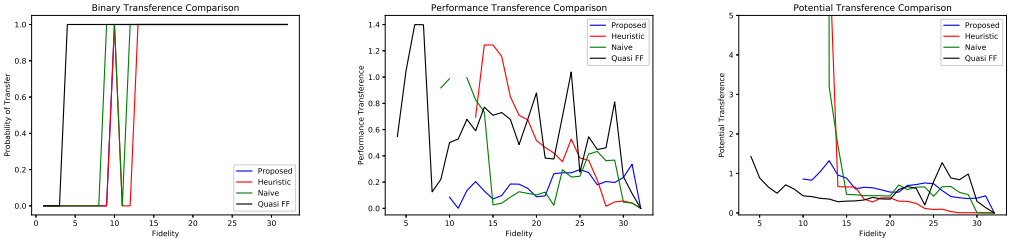


Figure D.11: System 10

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.8186 & -0.1802 & -0.7008 & -0.1891 \\ -0.9869 & -0.2928 & -0.2894 & -0.2236 \\ 0.6439 & 0.4446 & -0.0664 & -0.7716 \\ -0.6104 & 0.7932 & 0.1992 & 0.8569 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.2399 & 0.4364 & -0.6952 & 0.1399 \\ 0.5043 & -0.9518 & 0.5120 & -0.6010 \\ 0.1483 & -0.8640 & 0.0888 & 0.4992 \\ 0.8166 & 0.8802 & -0.0263 & -0.4942 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

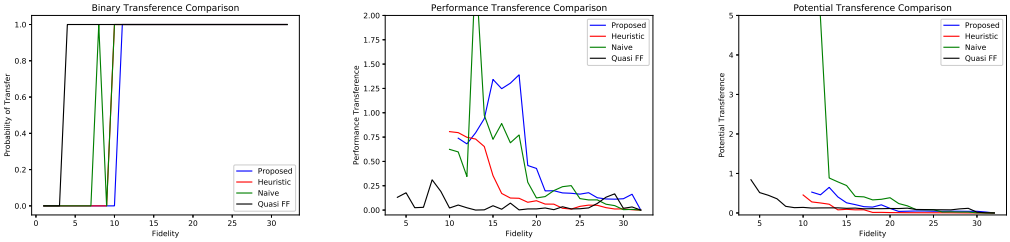


Figure D.12: System 11

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.1377 & -0.2194 & -0.7063 & -0.4527 \\ 0.1996 & -0.4399 & 0.6750 & 0.5802 \\ 0.5613 & -0.6534 & 0.8928 & 0.5947 \\ 0.3728 & 0.8672 & 0.5558 & -0.0151 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.3827 & -0.3314 & 0.3138 & -0.6142 \\ -0.0700 & 0.5067 & -0.9480 & -0.9146 \\ -0.2964 & -0.4816 & 0.7829 & 0.8165 \\ 0.3754 & -0.8241 & 0.6992 & -0.6527 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

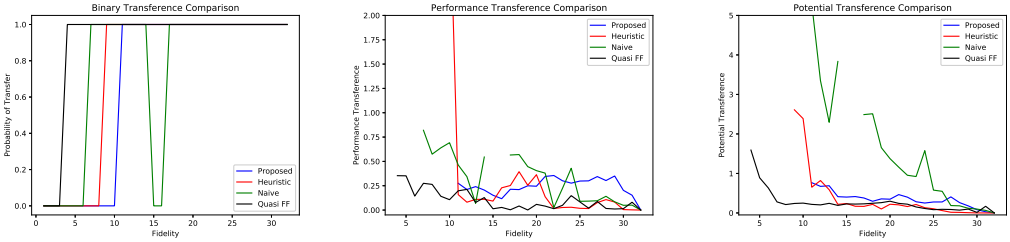


Figure D.13: System 12

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.5761 & -0.3287 & -0.7031 & -0.7621 \\ 0.7703 & -0.7362 & -0.8574 & 0.3523 \\ 0.3613 & -0.3870 & 0.5145 & -0.0042 \\ -0.6138 & 0.9110 & -0.3164 & 0.0572 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.1534 & 0.6365 & 0.6377 & -0.6061 \\ -0.6833 & 0.4880 & -0.7331 & -0.3484 \\ 0.4816 & -0.0672 & 0.6104 & 0.7204 \\ 0.7910 & 0.6643 & 0.7824 & 0.8301 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

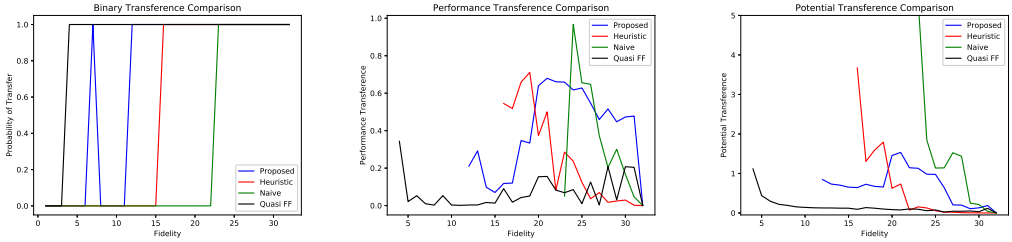


Figure D.14: System 13

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.7263 & 0.9000 & -0.0084 & 0.1725 \\ 0.6938 & -0.7155 & 0.6600 & -0.0035 \\ 0.5649 & -0.0050 & 0.4885 & 0.3627 \\ 0.7278 & -0.5683 & 0.3527 & 0.4148 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.9050 & -0.7606 & 0.1605 & -0.4503 \\ 0.1659 & -0.7230 & 0.8784 & -0.2626 \\ 0.2993 & -0.4946 & -0.9119 & 0.0756 \\ -0.4336 & -0.4413 & 0.8064 & 0.0204 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

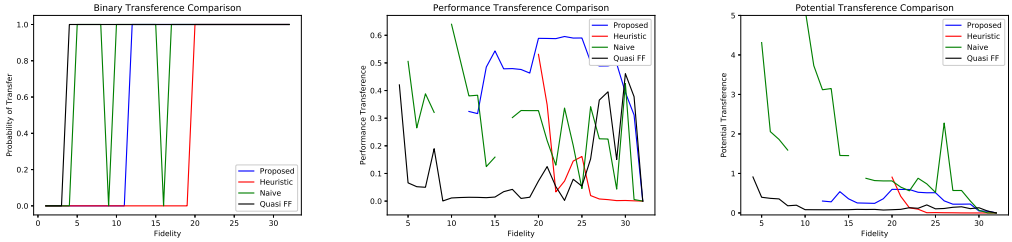


Figure D.15: System 14

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.5566 & -0.1963 & 0.1145 & 0.3264 \\ 0.7255 & -0.2176 & -0.2516 & -0.1972 \\ 0.9831 & 0.6729 & -0.3733 & -0.3909 \\ 0.5591 & 0.4675 & -0.9487 & -0.3149 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.3308 & -0.8868 & 0.7250 & -0.5487 \\ 0.1326 & 0.2249 & -0.5640 & 0.7372 \\ 0.2422 & -0.9883 & 0.5780 & -0.9262 \\ -0.7997 & -0.5119 & -0.0430 & -0.9012 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

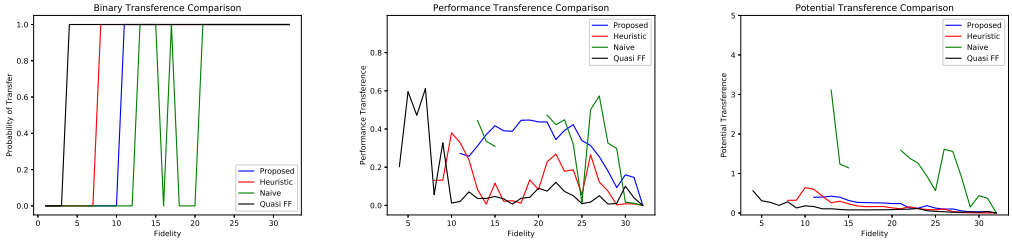


Figure D.16: System 15

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.2052 & -0.4665 & -0.3881 & 0.6739 \\ 0.5774 & 0.3897 & -0.1771 & -0.5184 \\ 0.2399 & -0.4778 & -0.4238 & -0.8042 \\ 0.1505 & 0.2926 & -0.2607 & 0.4290 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.7662 & 0.3335 & 0.0870 & -0.9076 \\ -0.2996 & 0.3720 & -0.8588 & 0.1677 \\ -0.9124 & -0.2906 & -0.0091 & 0.2784 \\ 0.2031 & 0.3629 & -0.6736 & -0.7525 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

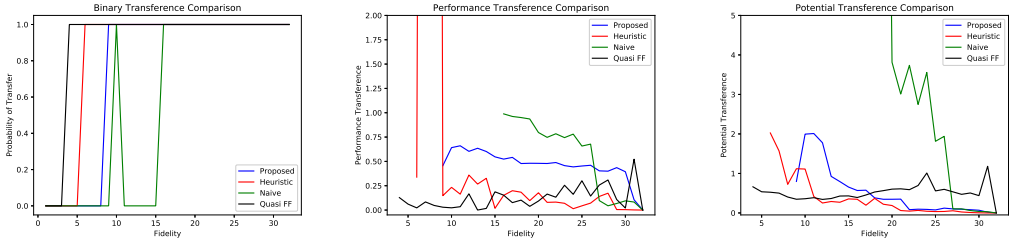


Figure D.17: System 16

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.3262 & 0.7410 & 0.2694 & -0.5687 \\ -0.0197 & 0.3070 & 0.5414 & 0.4708 \\ 0.4798 & 0.5910 & 0.8317 & 0.0698 \\ -0.9792 & -0.3231 & 0.2657 & 0.8067 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.5256 & 0.7568 & -0.8728 & -0.9380 \\ 0.4710 & -0.2649 & -0.5666 & 0.8311 \\ 0.5988 & -0.4145 & -0.6441 & -0.9967 \\ 0.0674 & -0.2075 & 0.3249 & 0.0347 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

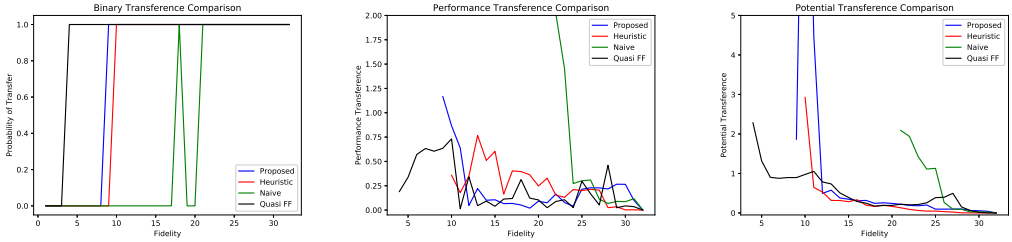


Figure D.18: System 17

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.1944 & 0.1372 & -0.8857 & 0.7021 \\ 0.9613 & 0.8408 & 0.3401 & 0.3696 \\ -0.3121 & 0.9721 & -0.2170 & -0.6533 \\ -0.1543 & 0.9495 & -0.2896 & 0.7713 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.9337 & -0.0403 & -0.6846 & -0.1229 \\ -0.3411 & -0.7272 & -0.7943 & -0.9515 \\ -0.6544 & 0.6261 & 0.6968 & -0.6740 \\ 0.1855 & -0.6269 & 0.6542 & 0.1743 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

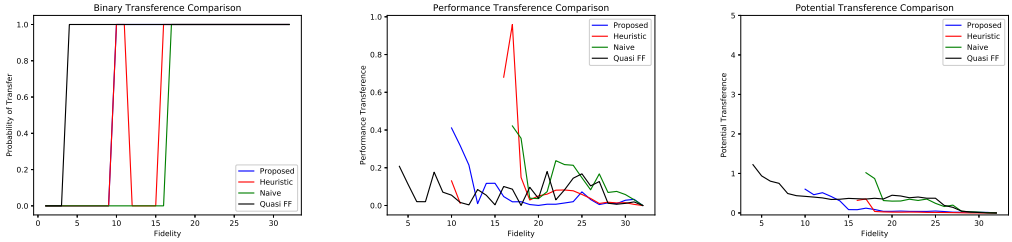


Figure D.19: System 18

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.1878 & 0.4942 & -0.0824 & -0.9110 \\ -0.6438 & 0.0330 & -0.7942 & 0.8921 \\ 0.4069 & 0.3336 & 0.4005 & -0.2164 \\ -0.8209 & 0.4568 & 0.1481 & 0.7207 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.9815 & 0.7807 & -0.5206 & 0.5015 \\ 0.9586 & 0.0330 & -0.5684 & 0.8162 \\ -0.5277 & -0.5503 & 0.8658 & -0.9220 \\ 0.7409 & -0.8285 & -0.6698 & -0.0431 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

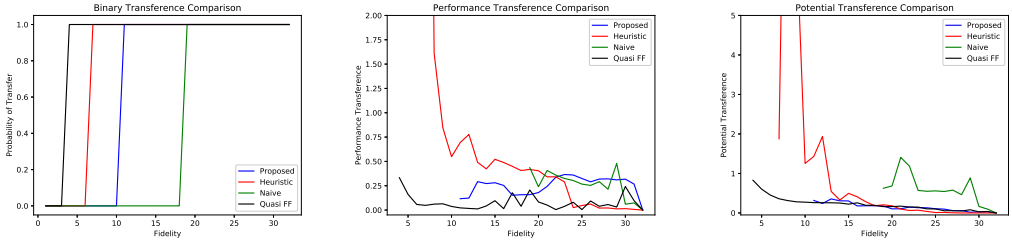


Figure D.20: System 19

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.3982 & 0.4779 & 0.4102 & -0.3796 \\ 0.6798 & -0.7181 & -0.5986 & 0.8683 \\ -0.1325 & 0.3500 & -0.3362 & 0.5836 \\ 0.1734 & 0.3455 & 0.7470 & 0.2914 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.5727 & -0.2019 & 0.8905 & -0.4438 \\ -0.0310 & 0.0636 & -0.5936 & -0.6722 \\ 0.5062 & 0.3696 & 0.6871 & -0.2395 \\ 0.1037 & 0.8178 & 0.2372 & -0.1712 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

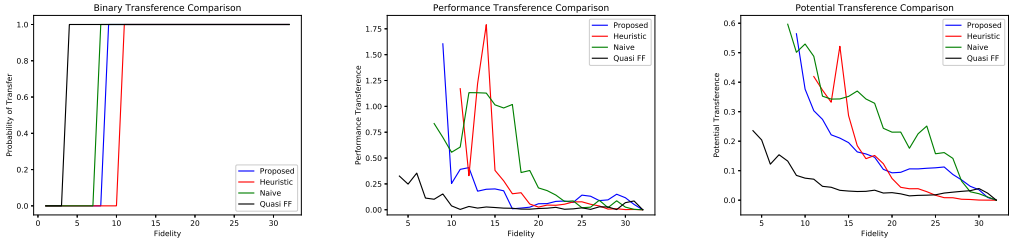


Figure D.21: System 20

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.3513 & 0.7172 & 0.6148 & 0.8002 \\ -0.9069 & -0.8571 & 0.9649 & -0.6862 \\ -0.6103 & 0.9504 & 0.9614 & 0.3986 \\ 0.6719 & -0.0540 & -0.0347 & -0.5874 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.5428 & -0.7914 & -0.3319 & 0.4190 \\ 0.3410 & 0.2237 & 0.8210 & 0.7542 \\ 0.1680 & -0.8735 & -0.6908 & 0.4014 \\ -0.8549 & 0.4928 & -0.1093 & -0.7895 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

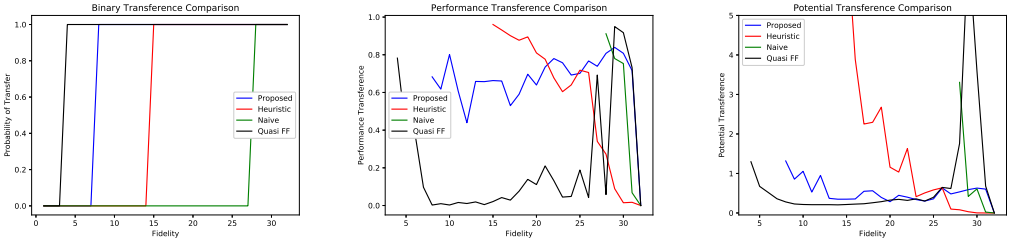


Figure D.22: System 21

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.1320 & 0.0993 & -0.4854 & 0.7458 \\ 0.2242 & -0.3225 & 0.4471 & 0.5586 \\ 0.3986 & -0.3793 & -0.7628 & 0.6913 \\ -0.1909 & -0.8552 & -0.5271 & -0.4029 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.5321 & -0.8647 & -0.4275 & -0.3822 \\ -0.0696 & 0.8869 & -0.3520 & 0.3303 \\ -0.4859 & 0.6489 & -0.1309 & -0.7262 \\ -0.6060 & 0.4558 & 0.7624 & -0.2204 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

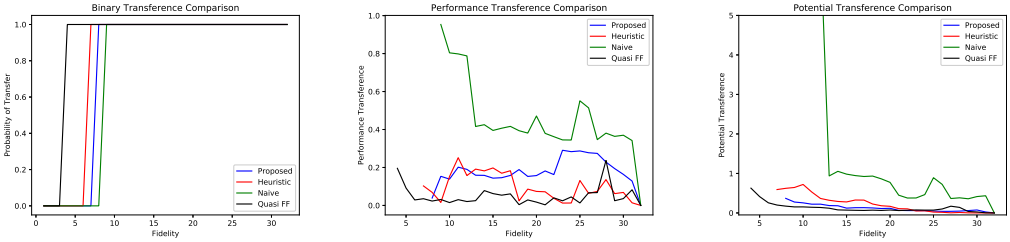


Figure D.23: System 22

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.1799 & -0.1579 & -0.9197 & 0.2015 \\ 0.9972 & -0.7863 & 0.6338 & -0.2156 \\ 0.7095 & 0.4872 & -0.1256 & -0.6562 \\ 0.4111 & 0.7051 & -0.7467 & -0.3627 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.2098 & 0.5511 & 0.4075 & 0.3483 \\ 0.2217 & -0.7659 & 0.2912 & -0.8229 \\ -0.8338 & -0.1320 & -0.2034 & -0.8320 \\ -0.9462 & 0.5981 & -0.0951 & -0.0010 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

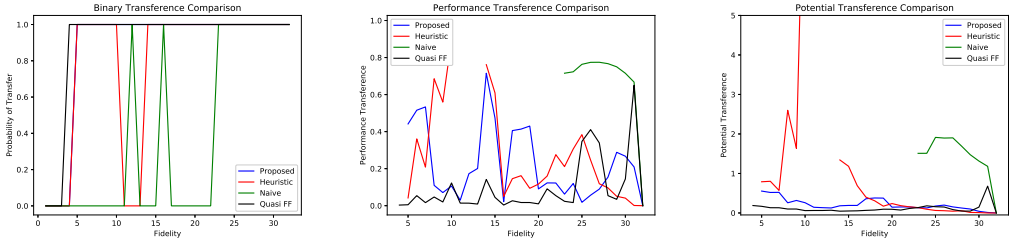


Figure D.24: System 23

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.1589 & 0.2728 & 0.9902 & -0.8628 \\ 0.3130 & 0.1698 & -0.6253 & -0.1015 \\ -0.0430 & 0.4434 & 0.9715 & -0.8452 \\ -0.2935 & 0.8057 & 0.0532 & 0.3111 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.6057 & -0.4107 & 0.8410 & 0.8537 \\ -0.1607 & 0.0942 & 0.0577 & -0.9700 \\ -0.5432 & -0.2644 & 0.4051 & -0.8546 \\ -0.4959 & 0.9095 & -0.1999 & -0.5982 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

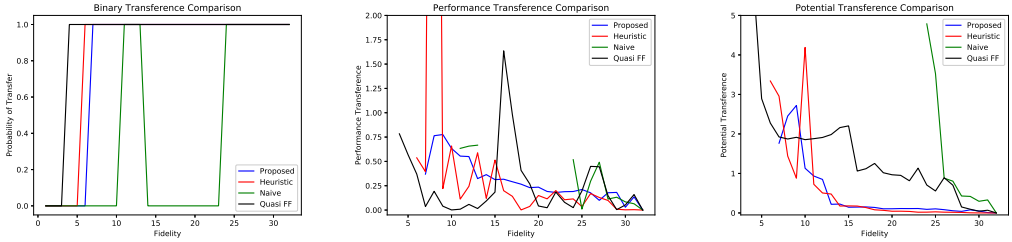


Figure D.25: System 24

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.5426 & 0.0937 & -0.8170 & 0.3904 \\ 0.1355 & -0.9668 & 0.7057 & 0.5096 \\ -0.0283 & -0.8811 & -0.1596 & -0.5646 \\ 0.6954 & 0.6180 & -0.3079 & -0.4546 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.5458 & 0.3549 & -0.3787 & -0.2985 \\ 0.1568 & -0.0376 & -0.5106 & -0.7747 \\ -0.1038 & 0.6887 & -0.6100 & 0.2989 \\ -0.8097 & -0.7490 & 0.8311 & -0.0356 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

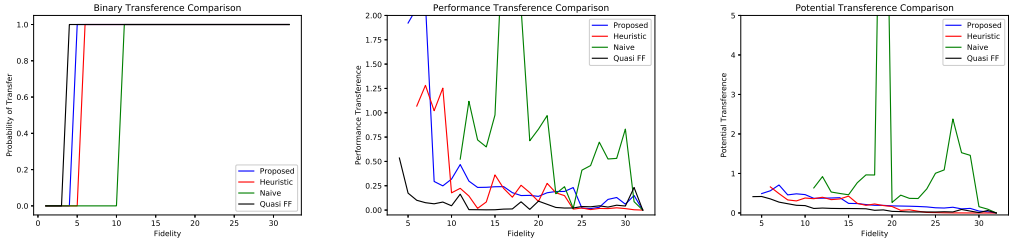


Figure D.26: System 25

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.4011 & 0.9527 & 0.2804 & 0.9408 \\ -0.2983 & 0.0690 & -0.4347 & 0.3179 \\ 0.2396 & -0.6680 & 0.4357 & -0.2366 \\ 0.2677 & 0.4567 & 0.8781 & -0.2051 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.1086 & 0.9682 & -0.1682 & 0.4161 \\ 0.7867 & -0.5126 & -0.6071 & 0.9021 \\ 0.9525 & 0.1957 & 0.4369 & -0.0453 \\ -0.3943 & -0.0766 & 0.0707 & -0.1621 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

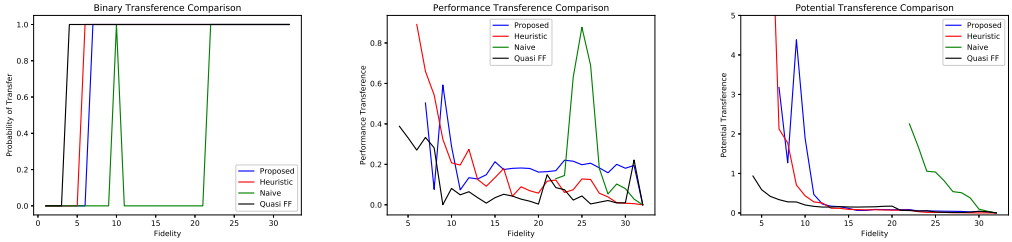


Figure D.27: System 26

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.7809 & 0.0313 & 0.5016 & 0.7733 \\ -0.0778 & 0.3544 & 0.6089 & -0.1831 \\ 0.2720 & -0.6010 & 0.4484 & 0.0382 \\ -0.7935 & -0.9783 & -0.8752 & 0.8733 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.2541 & 0.0402 & 0.2533 & -0.7266 \\ 0.4495 & -0.1974 & 0.0473 & -0.1906 \\ -0.1516 & -0.4939 & -0.5871 & 0.7005 \\ -0.1745 & 0.9453 & -0.1482 & 0.1772 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

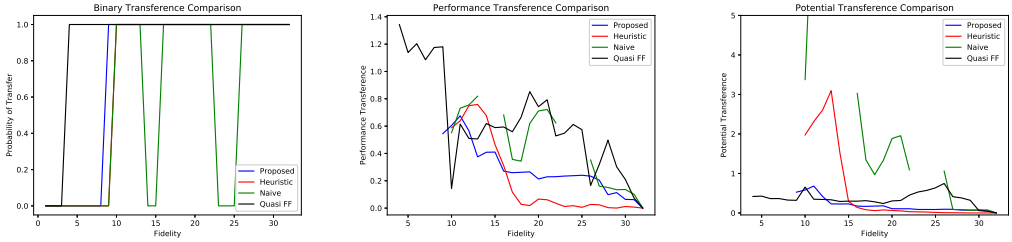


Figure D.28: System 27

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.3289 & -0.2445 & -0.6689 & -0.1773 \\ 0.4067 & 0.7271 & -0.7081 & 0.5827 \\ -0.9235 & 0.7243 & -0.5280 & -0.4214 \\ 0.3591 & -0.2445 & 0.3191 & 0.3900 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.8125 & 0.4791 & -0.9732 & -0.0942 \\ 0.3407 & -0.7110 & 0.1292 & 0.6778 \\ -0.9956 & 0.4353 & -0.7615 & 0.9611 \\ 0.8037 & 0.9503 & -0.1519 & -0.6239 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

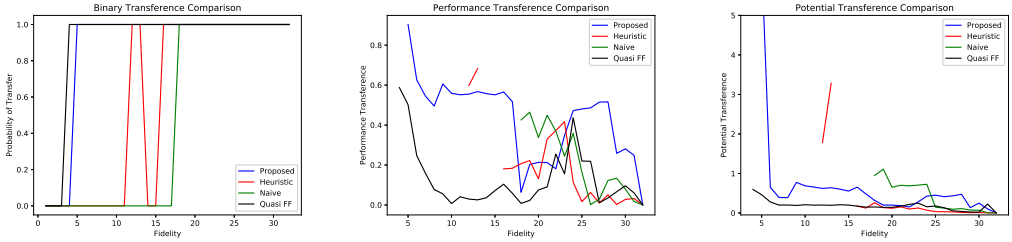


Figure D.29: System 28

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.1637 & -0.1060 & -0.4344 & 0.8614 \\ 0.8992 & 0.4541 & 0.4510 & 0.4281 \\ 0.8276 & 0.3960 & 0.7351 & 0.6273 \\ 0.3134 & -0.6811 & 0.1269 & 0.1475 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.7436 & 0.0815 & 0.4283 & 0.2210 \\ -0.1423 & 0.7358 & -0.2808 & 0.6276 \\ -0.6345 & -0.1738 & -0.2054 & 0.8976 \\ -0.4535 & 0.5617 & -0.7248 & -0.4737 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

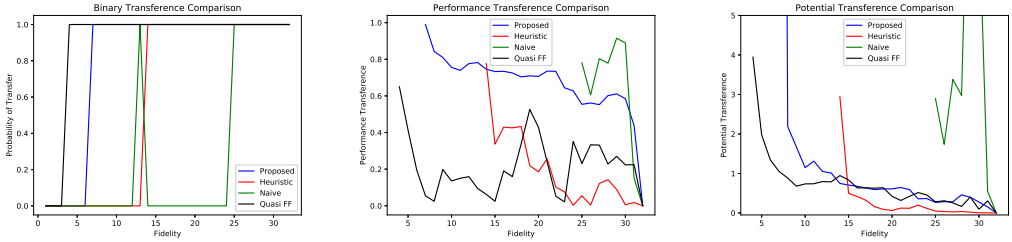


Figure D.30: System 29

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.5883 & 0.0628 & -0.5768 & -0.3762 \\ -0.4616 & 0.0523 & 0.8305 & -0.9916 \\ -0.4193 & -0.8828 & -0.3892 & 0.3953 \\ -0.4333 & -0.2255 & -0.7286 & 0.2545 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.9360 & -0.8304 & -0.8842 & -0.0167 \\ -0.0759 & -0.1309 & -0.9932 & 0.1790 \\ -0.4720 & -0.4767 & -0.9131 & -0.1626 \\ -0.4798 & -0.5065 & 0.5920 & 0.3184 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

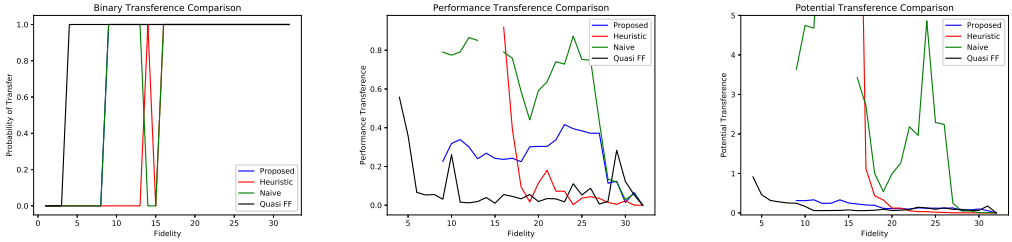


Figure D.31: System 30

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.9872 & 0.8610 & 0.8183 & 0.6793 \\ -0.2225 & -0.5985 & 0.9050 & -0.5786 \\ -0.0485 & -0.4812 & -0.0173 & -0.3853 \\ 0.7950 & 0.6018 & -0.2331 & 0.4181 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.6291 & 0.9477 & -0.1493 & 0.5087 \\ 0.8506 & 0.3575 & 0.1028 & -0.0556 \\ -0.6488 & -0.4561 & -0.4002 & -0.9487 \\ -0.3747 & -0.2110 & -0.8238 & -0.4188 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

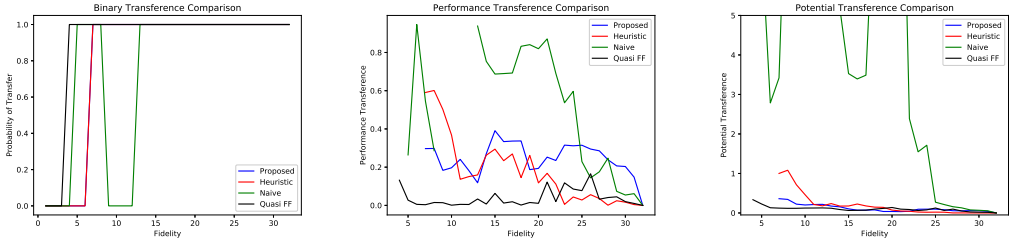


Figure D.32: System 31

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.7659 & 0.5348 & -0.8032 & 0.6970 \\ 0.5936 & 0.2540 & 0.1186 & -0.6038 \\ 0.2412 & 0.7464 & 0.0020 & -0.5182 \\ -0.4102 & -0.9911 & -0.2951 & 0.2797 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.3872 & -0.0960 & -0.9528 & -0.5480 \\ 0.5511 & -0.7685 & 0.9384 & -0.4724 \\ -0.7310 & -0.7139 & -0.2494 & 0.0424 \\ 0.7515 & 0.1497 & 0.6201 & -0.2492 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

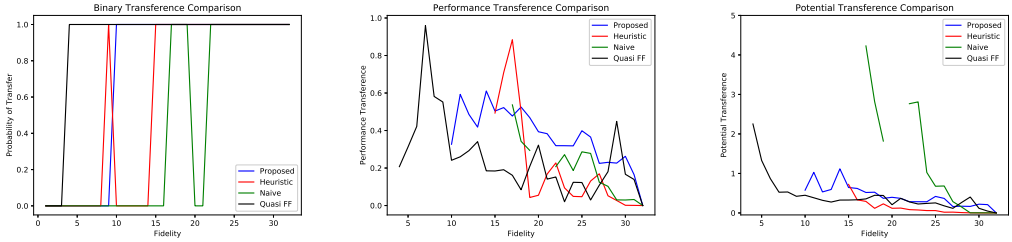


Figure D.33: System 32

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.9968 & -0.2200 & -0.8769 & 0.4577 \\ 0.5611 & -0.3568 & -0.2781 & 0.6175 \\ -0.3177 & -0.0229 & 0.4587 & -0.6391 \\ -0.2414 & -0.8692 & -0.3209 & 0.8295 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.7539 & -0.8530 & 0.0399 & 0.0396 \\ -0.1271 & 0.2368 & -0.6265 & 0.3914 \\ -0.9459 & 0.8520 & -0.7411 & 0.7012 \\ 0.6253 & 0.3676 & 0.0743 & -0.2747 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

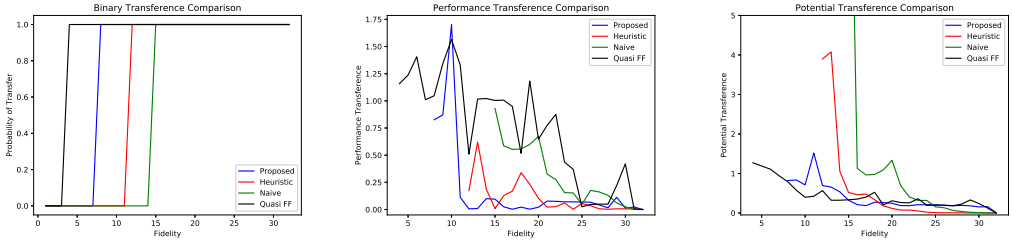


Figure D.34: System 33

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.6315 & -0.4906 & 0.6706 & -0.9021 \\ -0.8645 & -0.5654 & 0.2123 & 0.7471 \\ -0.3051 & 0.7476 & -0.4070 & -0.2218 \\ 0.0595 & -0.1466 & 0.4499 & -0.5188 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.3264 & 0.4999 & -0.5030 & -0.7696 \\ -0.1659 & -0.8277 & 0.8783 & 0.1323 \\ -0.7345 & 0.6622 & -0.4542 & -0.7934 \\ -0.7130 & 0.2463 & -0.3305 & -0.8899 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

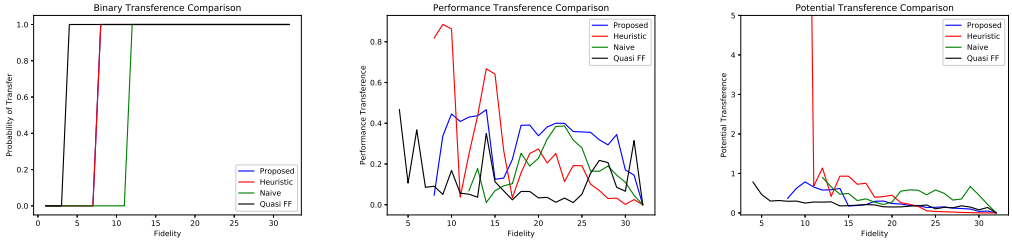


Figure D.35: System 34

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.2391 & 0.9672 & -0.6156 & -0.1249 \\ 0.9389 & 0.6802 & -0.3383 & -0.8622 \\ 0.8572 & -0.1192 & -0.5237 & 0.2694 \\ 0.8370 & 0.8838 & -0.6899 & -0.9607 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.4933 & -0.9882 & 0.9201 & -0.2858 \\ 0.1144 & -0.3276 & 0.6217 & -0.3918 \\ 0.4796 & -0.3609 & -0.0314 & 0.3976 \\ -0.0596 & -0.8587 & 0.1921 & 0.8697 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

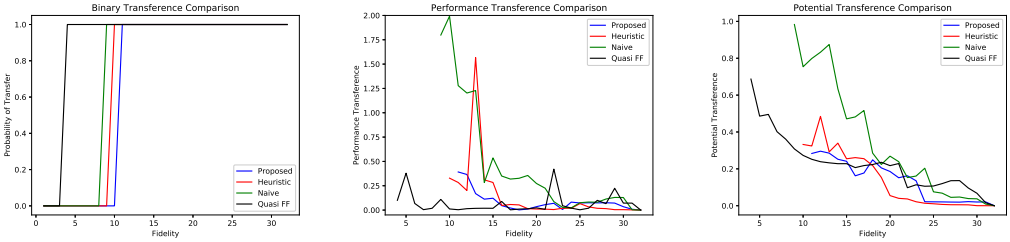


Figure D.36: System 35

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.7449 & 0.8981 & 0.7320 & 0.6384 \\ 0.8565 & 0.7062 & -0.5716 & -0.2407 \\ -0.8975 & -0.8237 & 0.2269 & -0.7574 \\ 0.5317 & -0.9681 & -0.4261 & 0.5966 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.9143 & -0.1683 & 0.9338 & 0.5315 \\ 0.0641 & -0.0629 & -0.2866 & 0.5326 \\ -0.7296 & -0.7901 & -0.8784 & 0.2357 \\ 0.4512 & -0.7352 & -0.2211 & -0.7885 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

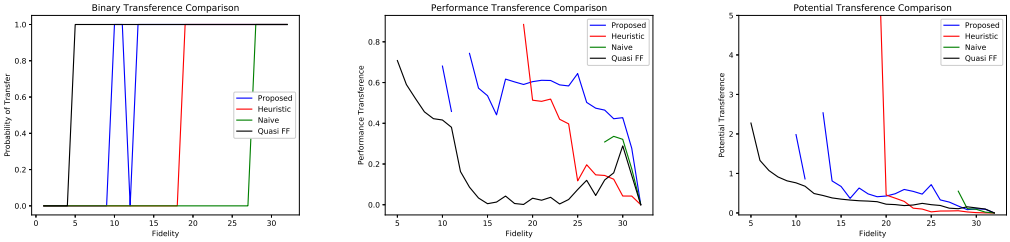


Figure D.37: System 36

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.3500 & -0.2577 & 0.5820 & -0.6260 \\ 0.4070 & -0.3234 & 0.4355 & -0.9878 \\ 0.6944 & -0.7816 & 0.3383 & 0.3070 \\ 0.0288 & -0.9242 & 0.9817 & -0.0630 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.4539 & -0.6529 & 0.2779 & -0.1838 \\ -0.8299 & 0.1463 & 0.9353 & 0.7089 \\ -0.9991 & -0.0868 & 0.5659 & 0.2796 \\ -0.4531 & -0.0934 & 0.3880 & -0.2340 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

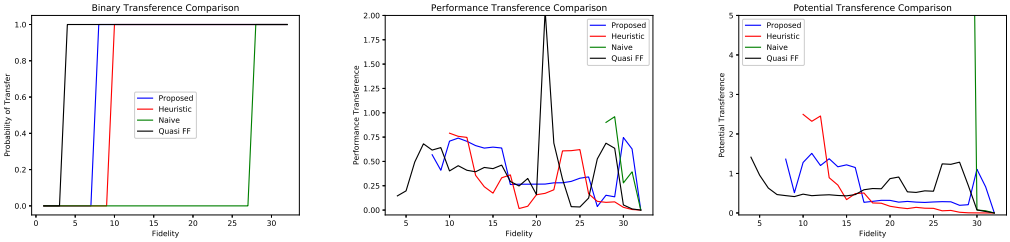


Figure D.38: System 37

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.5020 & -0.2176 & 0.0412 & -0.6612 \\ 0.1086 & 0.7679 & -0.1235 & 0.8516 \\ 0.0769 & -0.6735 & 0.2713 & 0.7789 \\ 0.5300 & 0.8576 & -0.2142 & -0.8448 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.2531 & 0.4316 & -0.5952 & 0.6165 \\ 0.2837 & -0.8892 & -0.7570 & 0.9231 \\ 0.5069 & 0.3966 & 0.6808 & 0.5459 \\ 0.8099 & -0.7672 & -0.0988 & 0.8611 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

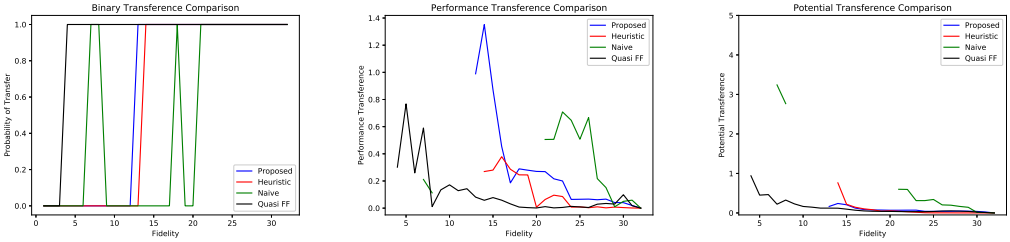


Figure D.39: System 38

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.7097 & -0.2150 & -0.1037 & -0.6401 \\ -0.7870 & -0.3505 & 0.3843 & -0.7556 \\ 0.4581 & -0.0923 & 0.3596 & 0.0468 \\ -0.0824 & -0.9312 & 0.7382 & -0.4223 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.9504 & -0.8165 & 0.8803 & 0.0128 \\ 0.2421 & -0.0594 & -0.7761 & -0.2319 \\ -0.2844 & -0.6644 & 0.8502 & -0.5742 \\ 0.0522 & -0.9436 & -0.1598 & 0.7467 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

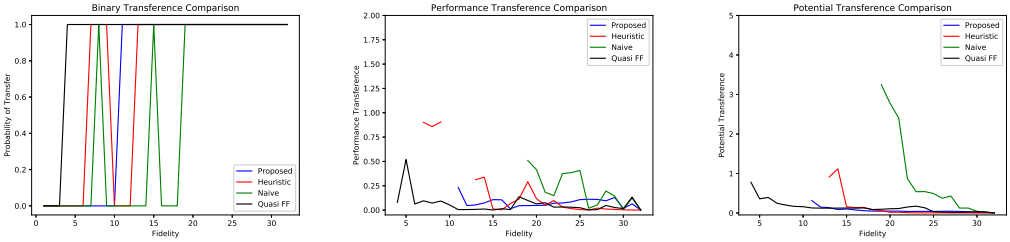


Figure D.40: System 39

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.4386 & 0.5555 & -0.2838 & -0.7723 \\ -0.2733 & -0.4958 & 0.0980 & 0.3558 \\ -0.9570 & 0.2880 & 0.0818 & -0.9399 \\ -0.5688 & -0.6952 & 0.4532 & -0.7123 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.2939 & -0.8011 & -0.1999 & -0.6747 \\ 0.4777 & -0.4608 & -0.6741 & 0.3776 \\ -0.8449 & -0.5829 & -0.3190 & -0.8601 \\ 0.5238 & -0.2567 & 0.9452 & -0.8738 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

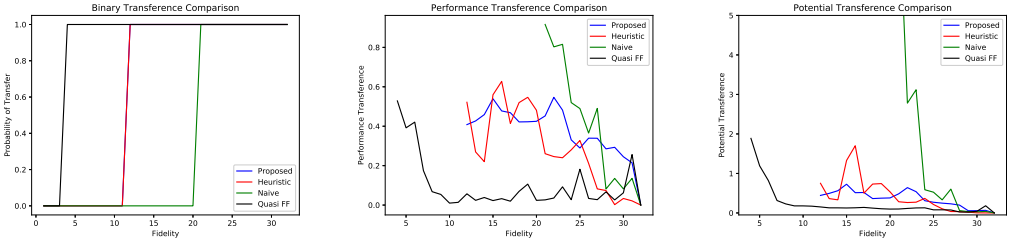


Figure D.41: System 40

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.3586 & 0.8755 & 0.0067 & 0.3057 \\ 0.1267 & -0.7117 & -0.5896 & 0.4322 \\ -0.4141 & 0.6949 & -0.6632 & 0.9627 \\ 0.9989 & 0.2059 & -0.5969 & 0.5085 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.8997 & 0.2447 & -0.9705 & -0.9856 \\ -0.5160 & 0.2211 & -0.1846 & -0.2815 \\ 0.1514 & 0.8106 & -0.6515 & 0.8456 \\ -0.8710 & -0.9677 & -0.4852 & -0.2112 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

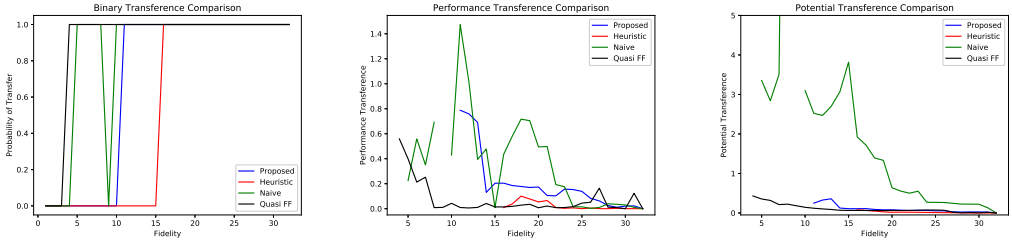


Figure D.42: System 41

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.9416 & 0.5054 & 0.7931 & -0.6628 \\ -0.6587 & 0.9000 & -0.3843 & 0.7403 \\ 0.7649 & -0.7193 & 0.0782 & 0.4682 \\ 0.1060 & -0.5584 & -0.3105 & -0.4253 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.2847 & -0.3872 & -0.0589 & -0.1141 \\ -0.1831 & 0.6829 & -0.9317 & -0.4167 \\ 0.9076 & 0.3713 & 0.4237 & 0.9823 \\ -0.0352 & 0.2463 & -0.6149 & 0.0831 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

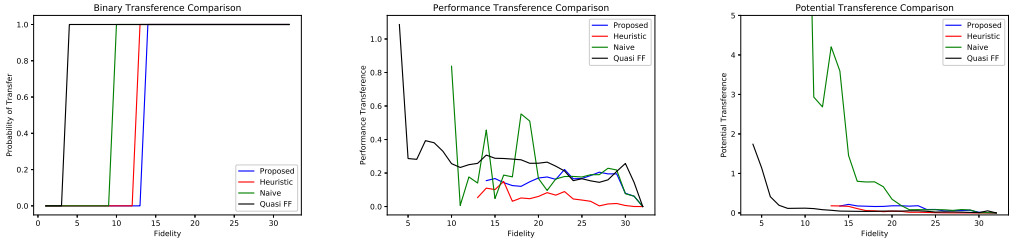


Figure D.43: System 42

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.5077 & -0.7149 & -0.2367 & -0.1380 \\ -0.3223 & 0.7627 & -0.4855 & -0.2655 \\ -0.3977 & 0.3119 & -0.7043 & 0.0000 \\ -0.3827 & 0.7030 & -0.5944 & 0.5333 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.2341 & 0.3950 & 0.9008 & 0.4680 \\ 0.1586 & 0.5399 & -0.2584 & -0.5134 \\ -0.0249 & -0.7250 & -0.4967 & 0.9375 \\ -0.5794 & 0.0784 & 0.7285 & 0.7052 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

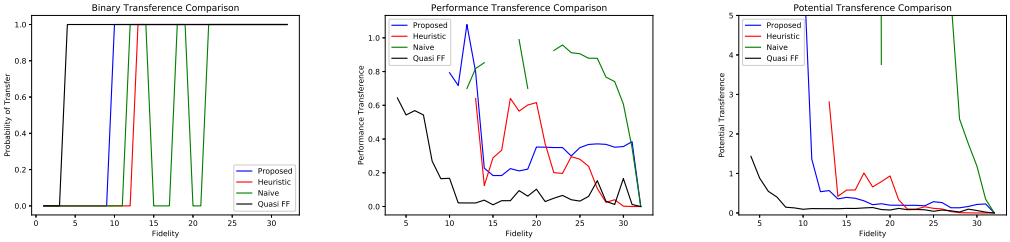


Figure D.44: System 43

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.6534 & -0.0283 & -0.4876 & -0.3889 \\ 0.7122 & 0.1057 & -0.8868 & 0.0217 \\ 0.7169 & -0.6836 & 0.9066 & -0.9740 \\ -0.4296 & 0.9569 & 0.0087 & -0.6176 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.3152 & -0.9380 & -0.9720 & 0.3822 \\ -0.7849 & -0.8127 & -0.6331 & 0.6646 \\ 0.5427 & -0.0847 & -0.9087 & 0.5532 \\ -0.2017 & 0.9002 & 0.0414 & -0.0345 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

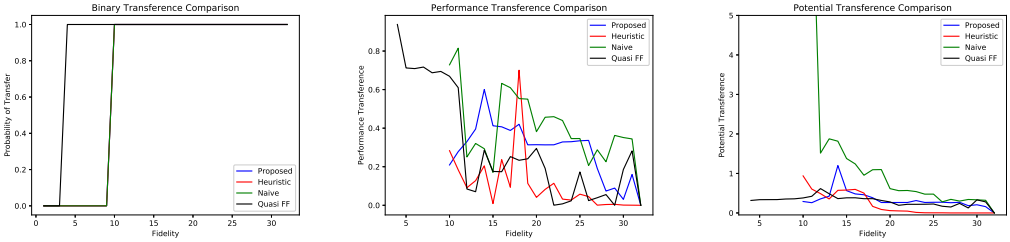


Figure D.45: System 44

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.4417 & 0.6043 & -0.1052 & -0.3189 \\ 0.2395 & 0.9964 & 0.1623 & 0.6596 \\ -0.5548 & 0.3648 & -0.4587 & -0.0432 \\ 0.6339 & 0.3751 & 0.5459 & -0.0921 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.5422 & 0.8966 & -0.4409 & 0.7609 \\ -0.6927 & 0.7312 & 0.5654 & 0.1907 \\ -0.6014 & -0.9407 & 0.2857 & -0.7042 \\ -0.3545 & -0.7328 & 0.8993 & 0.4669 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

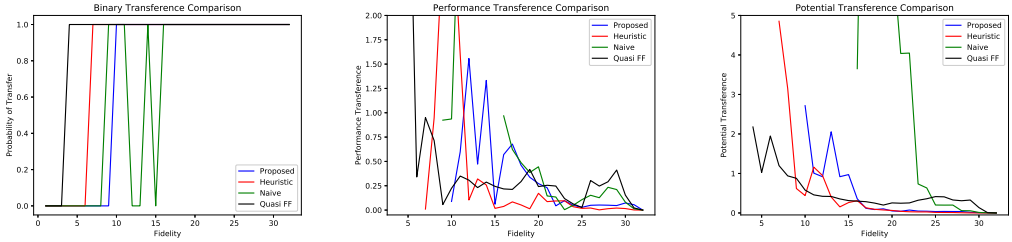


Figure D.46: System 45

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.9987 & 0.4291 & -0.3980 & 0.1126 \\ 0.6880 & 0.1131 & -0.4422 & 0.2177 \\ 0.7844 & 0.9077 & -0.3385 & 0.2325 \\ 0.4140 & 0.3332 & 0.0748 & 0.6805 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.2987 & -0.3505 & 0.2024 & -0.9530 \\ -0.0588 & 0.6015 & 0.3758 & -0.6393 \\ 0.1635 & 0.5874 & 0.8432 & 0.2883 \\ 0.5864 & -0.9161 & 0.5949 & 0.3374 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

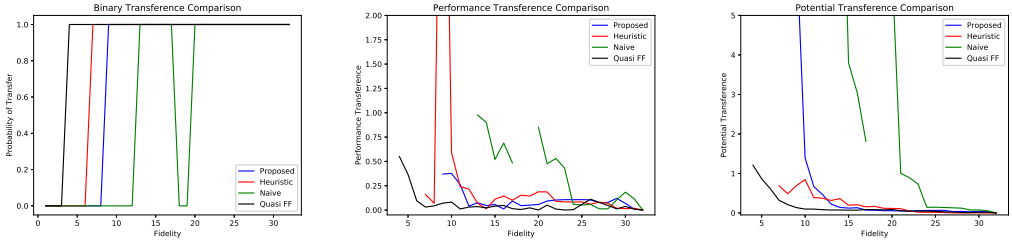


Figure D.47: System 46

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} -0.9917 & 0.8946 & -0.7564 & 0.8525 \\ 0.3739 & 0.0348 & -0.5081 & 0.9870 \\ 0.4869 & 0.7678 & 0.1511 & -0.0804 \\ 0.8170 & -0.9661 & 0.4385 & 0.5244 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.0019 & -0.6543 & -0.8890 & 0.3969 \\ -0.5285 & -0.5348 & -0.9081 & 0.0156 \\ -0.8397 & -0.2016 & 0.3003 & -0.4358 \\ 0.3420 & 0.4120 & 0.6718 & 0.3861 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

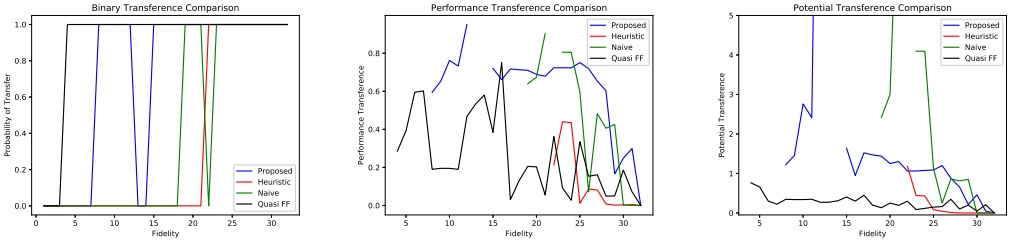


Figure D.48: System 47

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.2223 & -0.1248 & 0.6926 & 0.3998 \\ 0.4849 & -0.6825 & 0.3268 & 0.9253 \\ -0.9234 & -0.4609 & -0.8437 & -0.1324 \\ -0.6521 & -0.7919 & -0.3149 & 0.8063 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.6530 & 0.7171 & -0.2095 & 0.6343 \\ 0.3664 & 0.8799 & -0.1181 & -0.4926 \\ 0.1847 & 0.4734 & -0.9347 & 0.5036 \\ -0.4580 & 0.2532 & 0.4094 & -0.6198 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

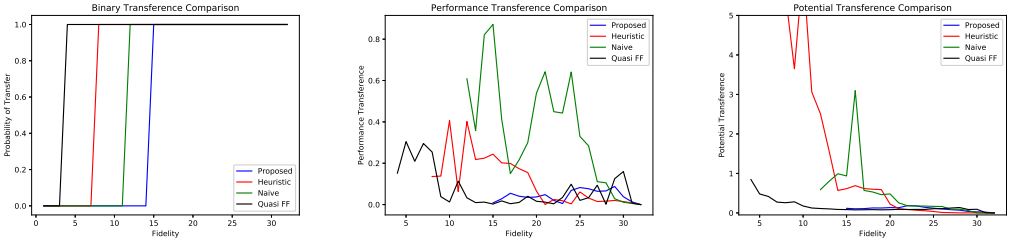


Figure D.49: System 48

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.0249 & -0.2778 & -0.1375 & 0.3957 \\ -0.7114 & 0.8979 & 0.4312 & -0.1431 \\ 0.6023 & -0.0188 & -0.3681 & -0.0412 \\ -0.4783 & 0.9976 & 0.7386 & 0.5197 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0.2848 & -0.2405 & 0.2602 & -0.9205 \\ -0.8245 & 0.2444 & 0.6979 & 0.8130 \\ 0.1506 & 0.8927 & -0.2429 & -0.8106 \\ -0.4560 & 0.3493 & 0.0733 & -0.7505 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

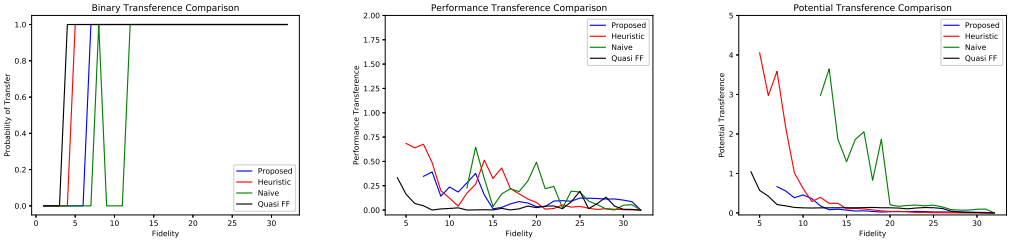
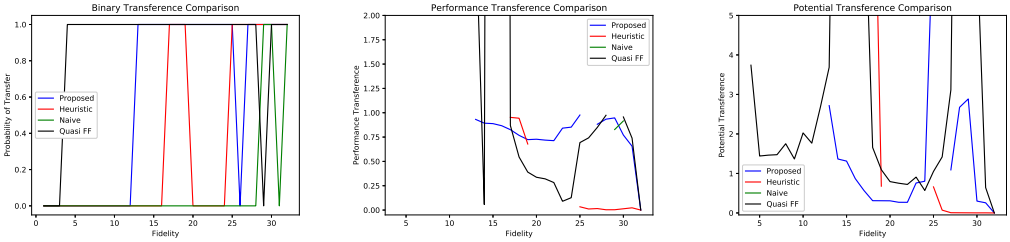


Figure D.50: System 49

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0.0516 & 0.2641 & 0.6370 & -0.6313 \\ -0.9330 & -0.9955 & -0.1723 & 0.2344 \\ 0.8490 & 0.0408 & 0.8231 & -0.4369 \\ -0.5574 & -0.9444 & 0.0900 & -0.8855 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.6445 & -0.9905 & 0.6271 & 0.8399 \\ 0.4161 & 0.8585 & -0.9086 & 0.6337 \\ 0.3683 & 0.9775 & -0.3207 & -0.6677 \\ 0.7052 & 0.8984 & 0.9852 & 0.4924 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

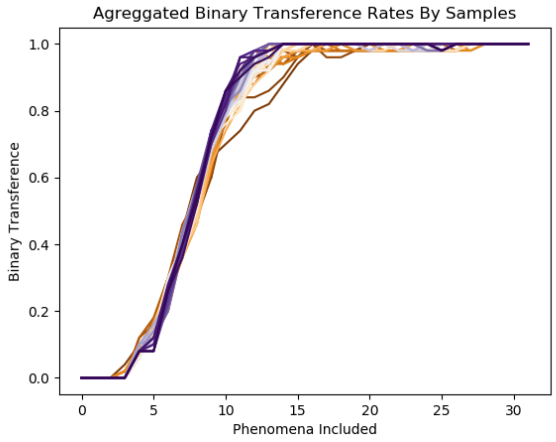


D.2 Experiment 2: Effects of Sampling Distribution

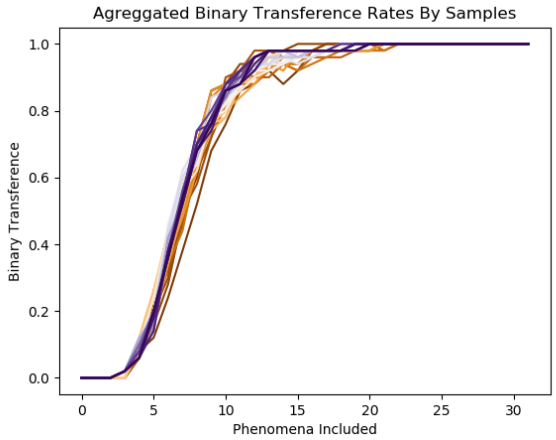
D.2.1 Transference Curves By Number of Samples

Section 5.2.2 discussed the major results for the sampling distribution experimentation. These experiments used the same truth systems as defined in Section D.1. In order to evaluate area under the transference curves, a new curve for each sampling density and distribution was generated. These are shown below in Figure D.51, Figure D.52, and Figure D.53, broken up by distribution and transference metric.

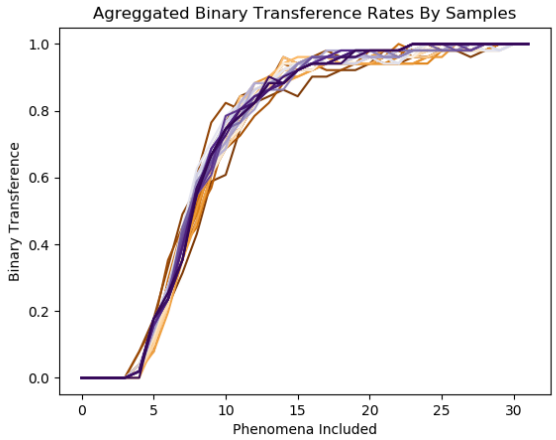
The triangular distribution is clearly the worst of the possible distributions for this class of systems, the criticality measures derived from it yield lower rates of transference across all but the highest fidelity systems. The difference between the uniform and representative distribution is smaller, as it is more of a direct tradeoff. The uniform distribution leads to better models at the very low end of fidelity, but slightly worse models for moderate fidelity levels. Depending on computational capabilities, either method may be preferable. However, the representative distribution did lead to better potential transference curves for this class of systems. As such, it is likely to be a fairly robust choice as a starting distribution until more information about the specific system of interest is not known.



(a) Representative Distribution

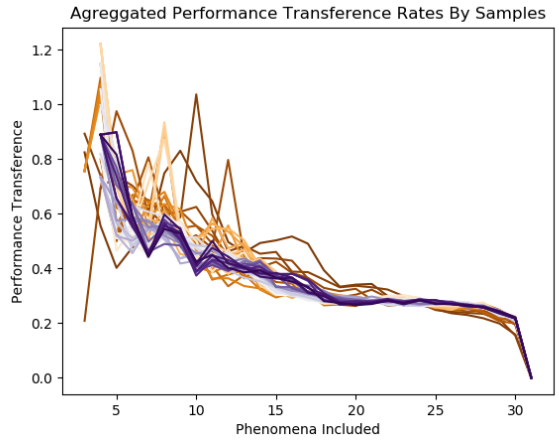


(b) Uniform Distribution

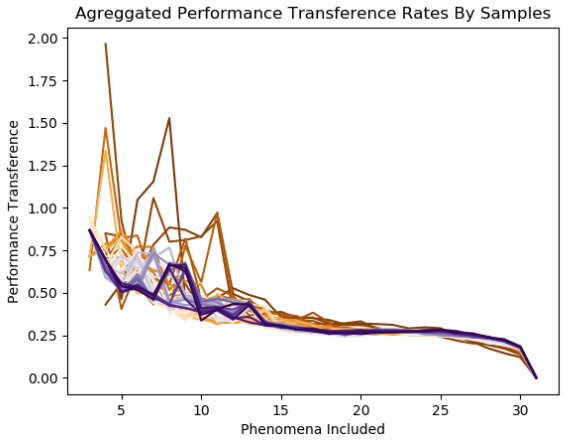


(c) Triangular Distribution

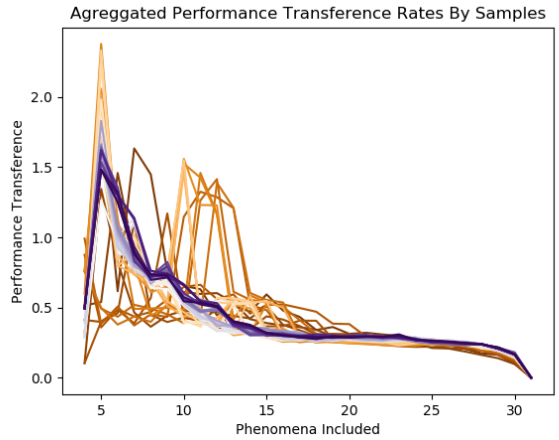
Figure D.51: Comparison of all Binary Transference curves for different sampling distributions and number of samples. Each of these curves was aggregated over the 50 linear systems discussed in Section D.1. Number of samples ranges from 200 (dark orange) to 10,000 (dark purple) in increments of 200.



(a) Representative Distribution

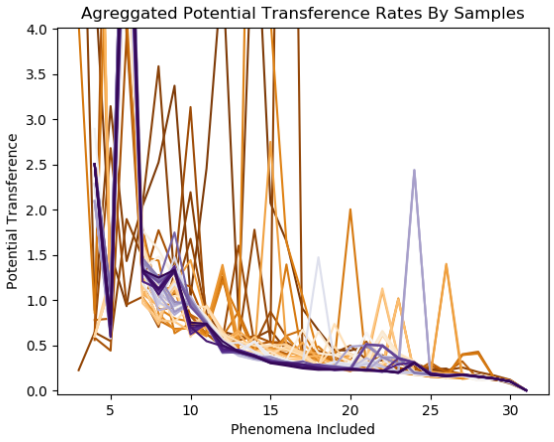


(b) Uniform Distribution

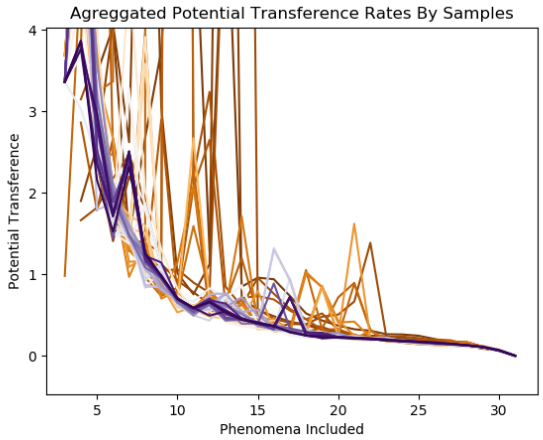


(c) Triangular Distribution

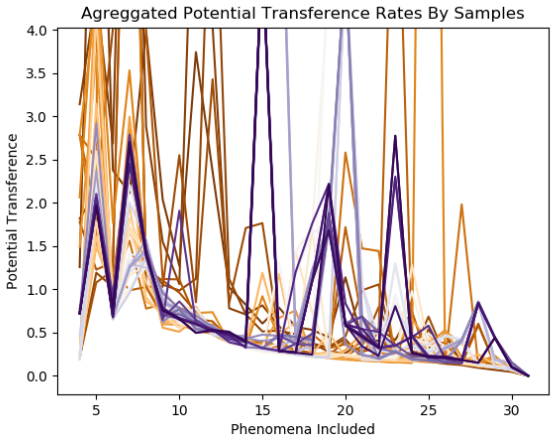
Figure D.52: Comparison of all Performance Transference curves for different sampling distributions and number of samples. Each of these curves was aggregated over the 50 linear systems discussed in Section D.1. Number of samples ranges from 200 (dark orange) to 10,000 (dark purple) in increments of 200.



(a) Representative Distribution



(b) Uniform Distribution



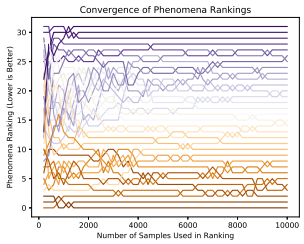
(c) Triangular Distribution

Figure D.53: Comparison of all Potential Transference curves for different sampling distributions and number of samples. Each of these curves was aggregated over the 50 linear systems discussed in Section D.1. Number of samples ranges from 200 (dark orange) to 10,000 (dark purple) in increments of 200.

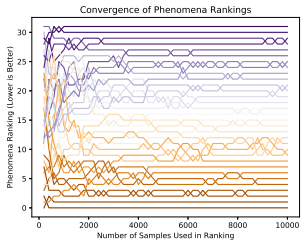
D.2.2 Individual System Phenomena Convergence

For this experiment, the convergence of phenomena rankings for each distribution type was also considered. For each system, one ordered set of 10,000 simplifications was sampled according to each of the three alternative methods. Within a set of simplifications, the method was applied multiple times, drawing an increasing number of simplifications from the set. First, 200 simplifications were used with the proposed method to determine criticality ranks. Then, in steps of 200, the number of samples used was increased and criticality ranks determined again until all 10,000 samples had been used. The resulting rankings are shown below for each of the three distributions discussed in Section 5.2.2 for all 50 individual systems.

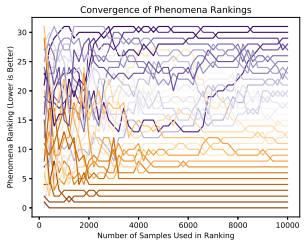
Figure D.54: System 0



(a) Representative

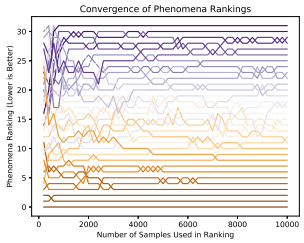


(b) Uniform Fidelity

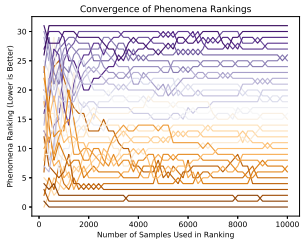


(c) Triangular Fidelity

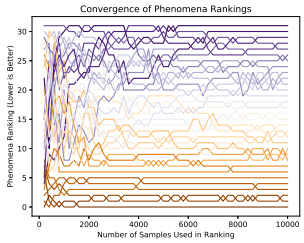
Figure D.55: System 1



(a) Representative

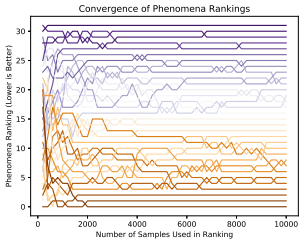


(b) Uniform Fidelity

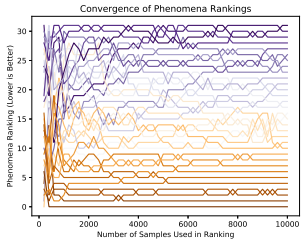


(c) Triangular Fidelity

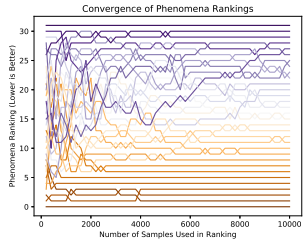
Figure D.56: System 2



(a) Representative

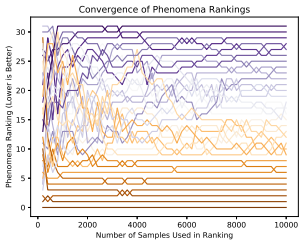


(b) Uniform Fidelity

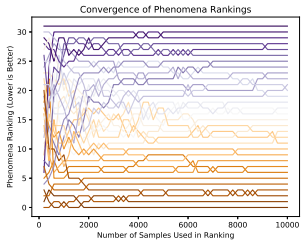


(c) Triangular Fidelity

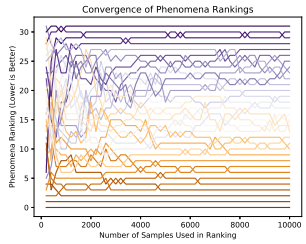
Figure D.57: System 3



(a) Representative

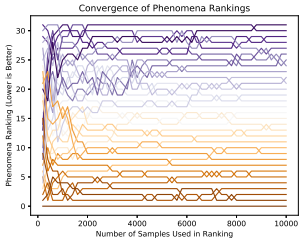


(b) Uniform Fidelity

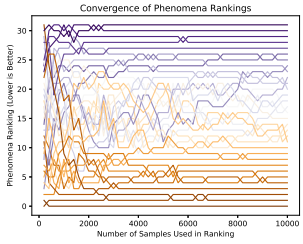


(c) Triangular Fidelity

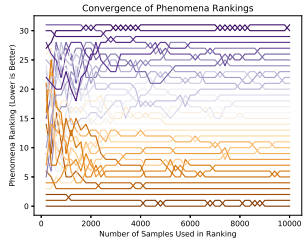
Figure D.58: System 4



(a) Representative

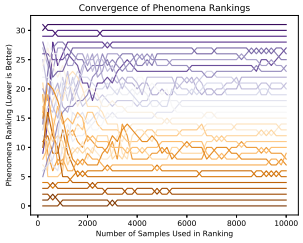


(b) Uniform Fidelity

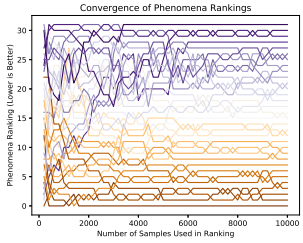


(c) Triangular Fidelity

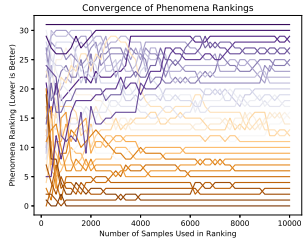
Figure D.59: System 5



(a) Representative

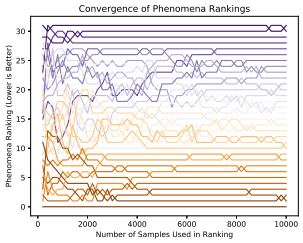


(b) Uniform Fidelity

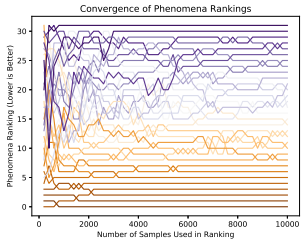


(c) Triangular Fidelity

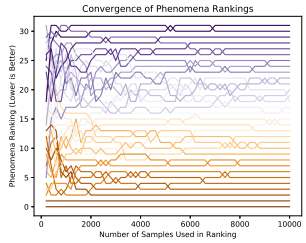
Figure D.60: System 6



(a) Representative

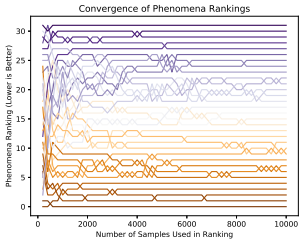


(b) Uniform Fidelity

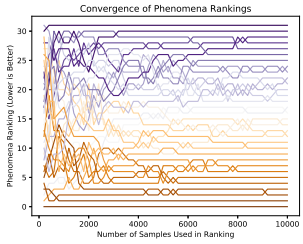


(c) Triangular Fidelity

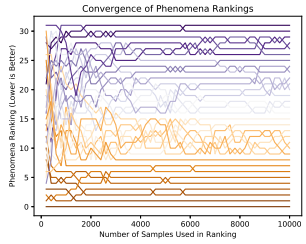
Figure D.61: System 7



(a) Representative

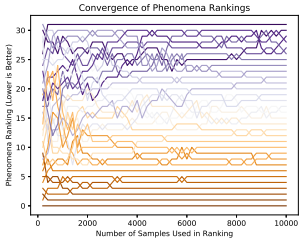


(b) Uniform Fidelity

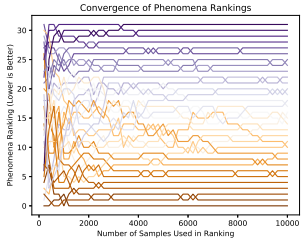


(c) Triangular Fidelity

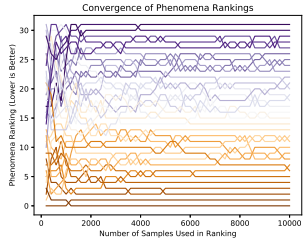
Figure D.62: System 8



(a) Representative

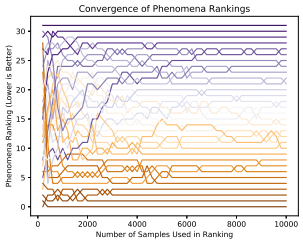


(b) Uniform Fidelity

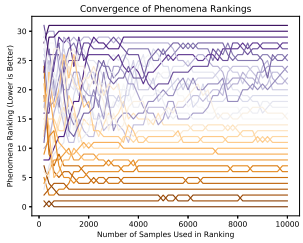


(c) Triangular Fidelity

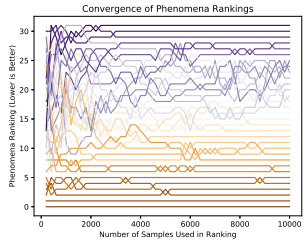
Figure D.63: System 9



(a) Representative

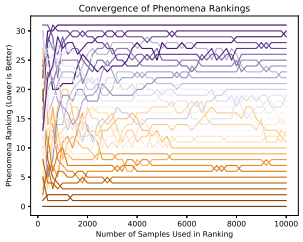


(b) Uniform Fidelity

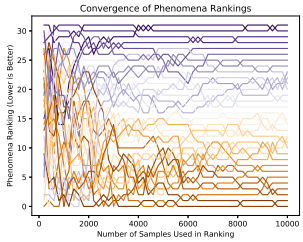


(c) Triangular Fidelity

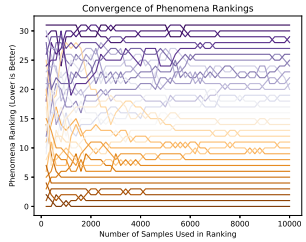
Figure D.64: System 10



(a) Representative

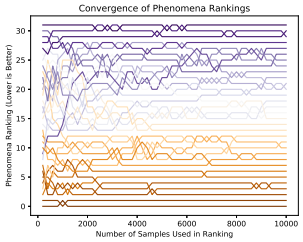


(b) Uniform Fidelity

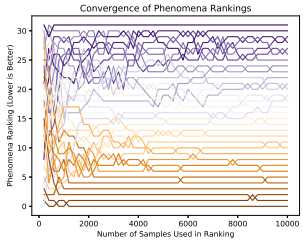


(c) Triangular Fidelity

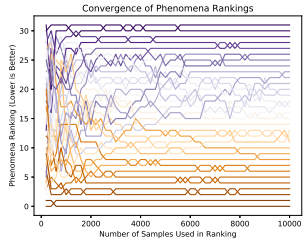
Figure D.65: System 11



(a) Representative

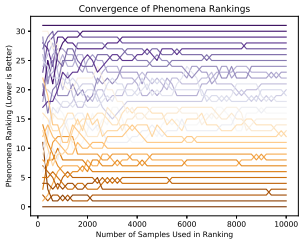


(b) Uniform Fidelity

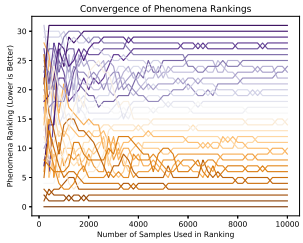


(c) Triangular Fidelity

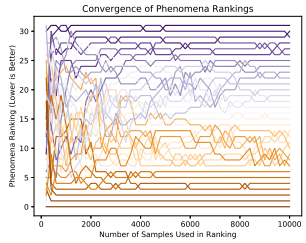
Figure D.66: System 12



(a) Representative

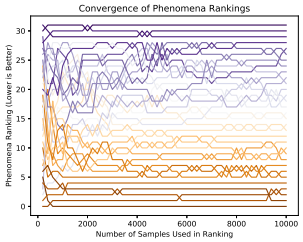


(b) Uniform Fidelity

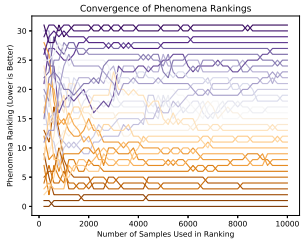


(c) Triangular Fidelity

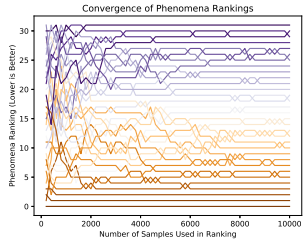
Figure D.67: System 13



(a) Representative

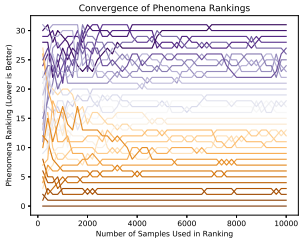


(b) Uniform Fidelity

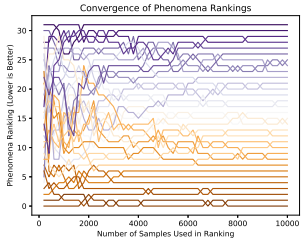


(c) Triangular Fidelity

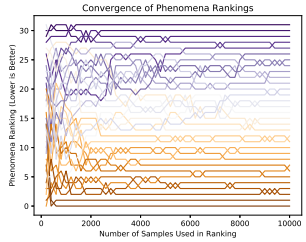
Figure D.68: System 14



(a) Representative

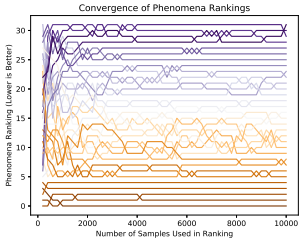


(b) Uniform Fidelity

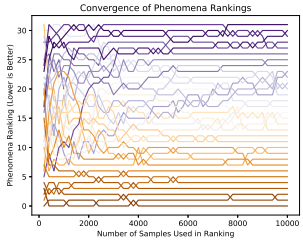


(c) Triangular Fidelity

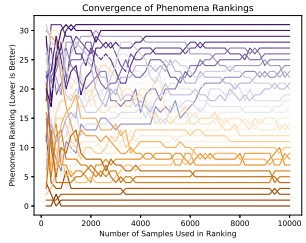
Figure D.69: System 15



(a) Representative

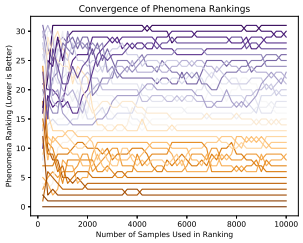


(b) Uniform Fidelity

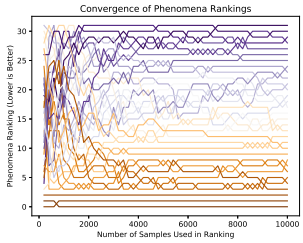


(c) Triangular Fidelity

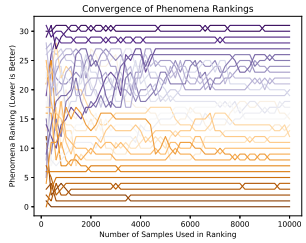
Figure D.70: System 16



(a) Representative

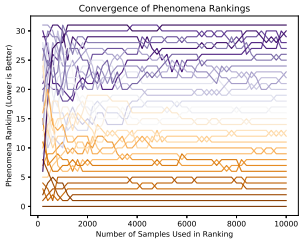


(b) Uniform Fidelity

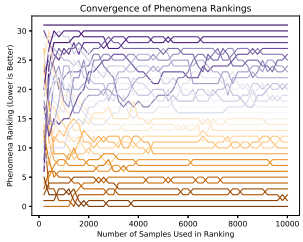


(c) Triangular Fidelity

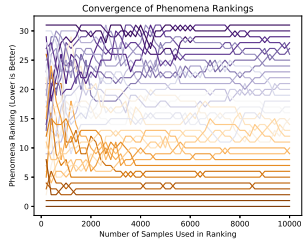
Figure D.71: System 17



(a) Representative

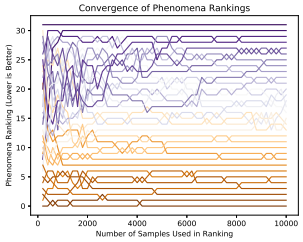


(b) Uniform Fidelity

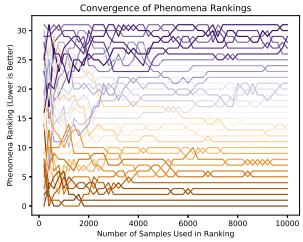


(c) Triangular Fidelity

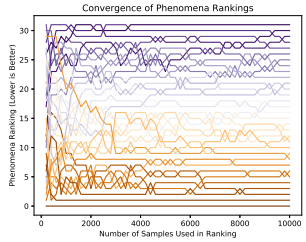
Figure D.72: System 18



(a) Representative

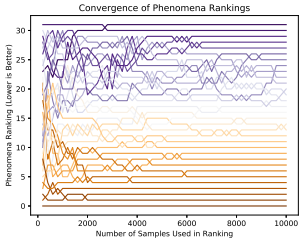


(b) Uniform Fidelity

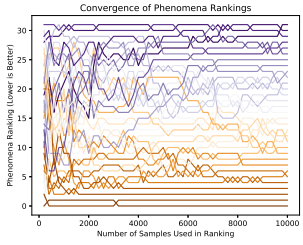


(c) Triangular Fidelity

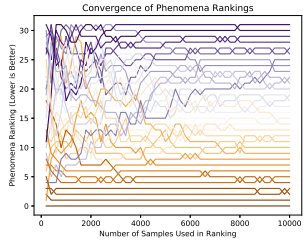
Figure D.73: System 19



(a) Representative

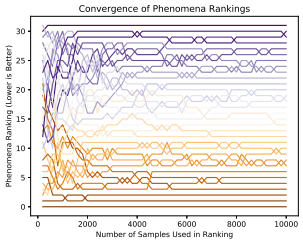


(b) Uniform Fidelity

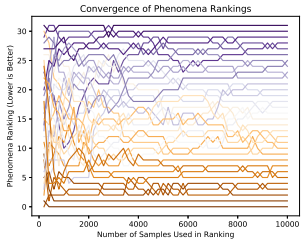


(c) Triangular Fidelity

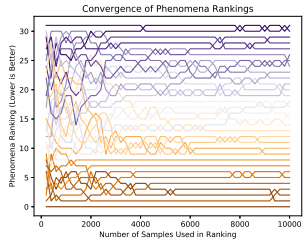
Figure D.74: System 20



(a) Representative

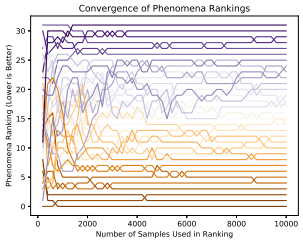


(b) Uniform Fidelity

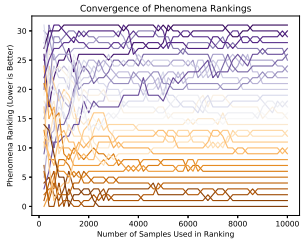


(c) Triangular Fidelity

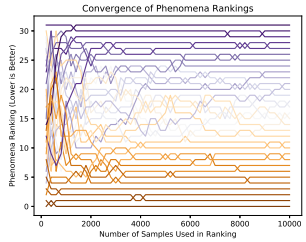
Figure D.75: System 21



(a) Representative

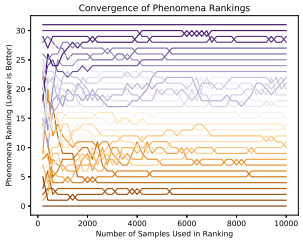


(b) Uniform Fidelity

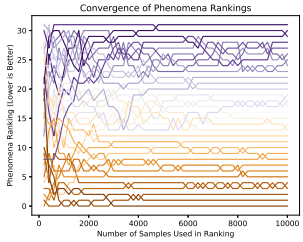


(c) Triangular Fidelity

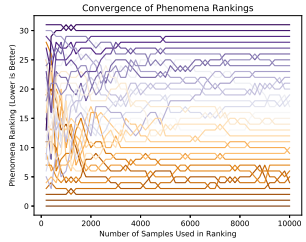
Figure D.76: System 22



(a) Representative

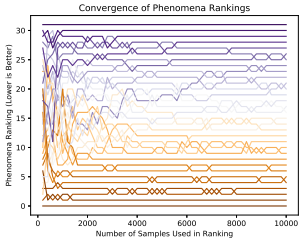


(b) Uniform Fidelity

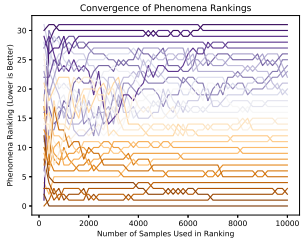


(c) Triangular Fidelity

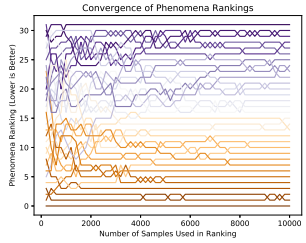
Figure D.77: System 23



(a) Representative

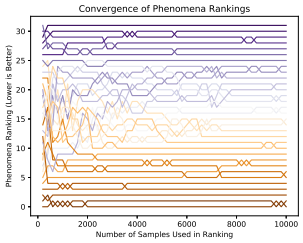


(b) Uniform Fidelity

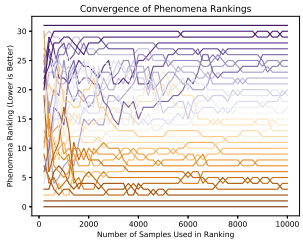


(c) Triangular Fidelity

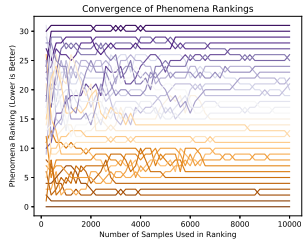
Figure D.78: System 24



(a) Representative

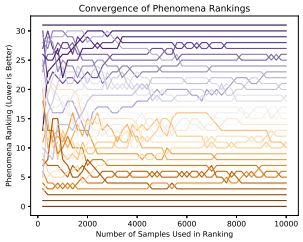


(b) Uniform Fidelity

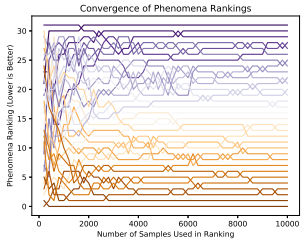


(c) Triangular Fidelity

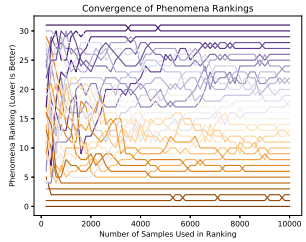
Figure D.79: System 25



(a) Representative

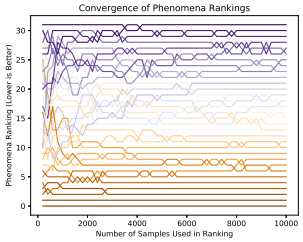


(b) Uniform Fidelity

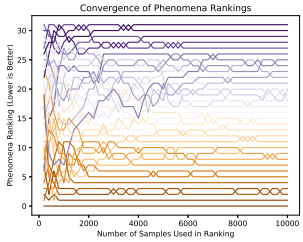


(c) Triangular Fidelity

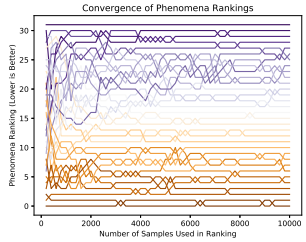
Figure D.80: System 26



(a) Representative

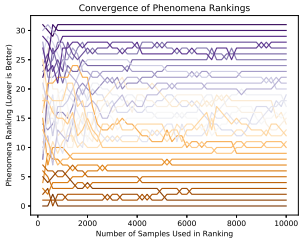


(b) Uniform Fidelity

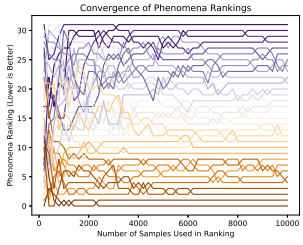


(c) Triangular Fidelity

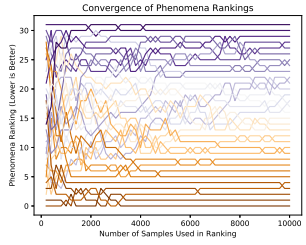
Figure D.81: System 27



(a) Representative

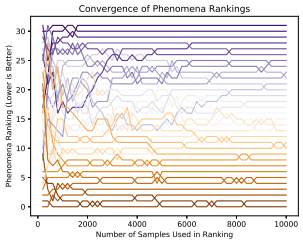


(b) Uniform Fidelity

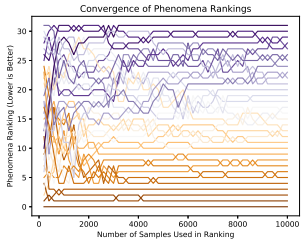


(c) Triangular Fidelity

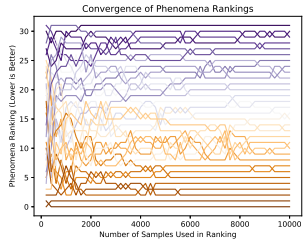
Figure D.82: System 28



(a) Representative

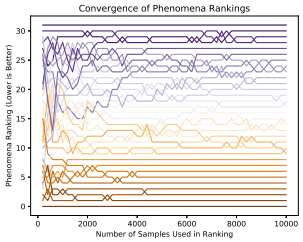


(b) Uniform Fidelity

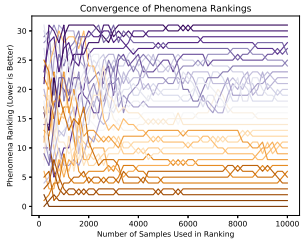


(c) Triangular Fidelity

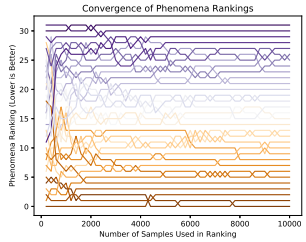
Figure D.83: System 29



(a) Representative

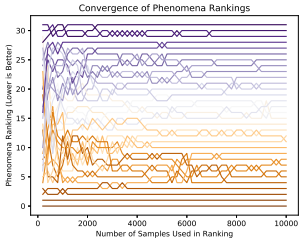


(b) Uniform Fidelity

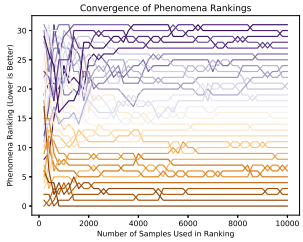


(c) Triangular Fidelity

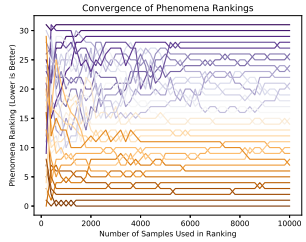
Figure D.84: System 30



(a) Representative

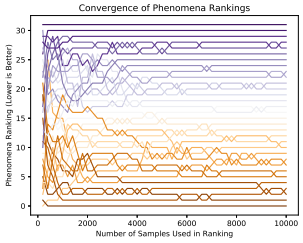


(b) Uniform Fidelity

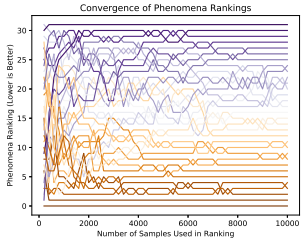


(c) Triangular Fidelity

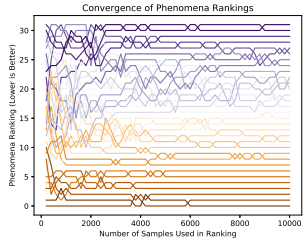
Figure D.85: System 31



(a) Representative

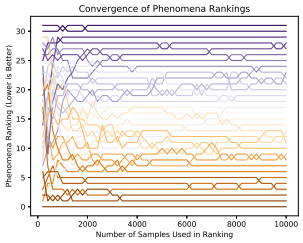


(b) Uniform Fidelity

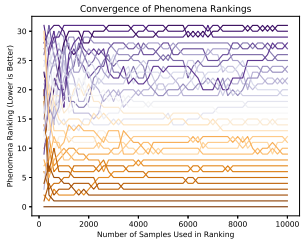


(c) Triangular Fidelity

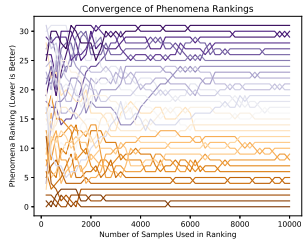
Figure D.86: System 32



(a) Representative

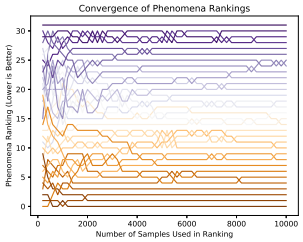


(b) Uniform Fidelity

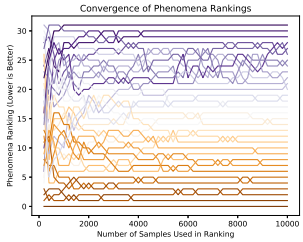


(c) Triangular Fidelity

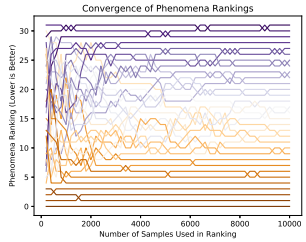
Figure D.87: System 33



(a) Representative

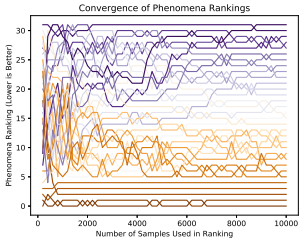


(b) Uniform Fidelity

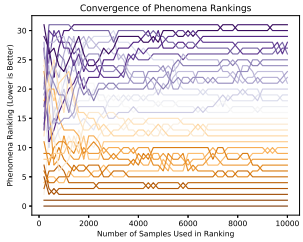


(c) Triangular Fidelity

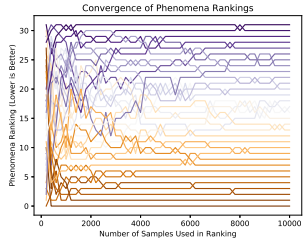
Figure D.88: System 34



(a) Representative

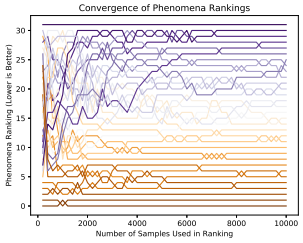


(b) Uniform Fidelity

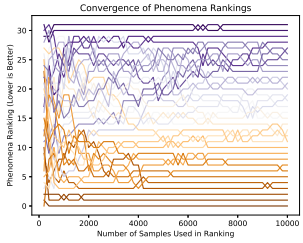


(c) Triangular Fidelity

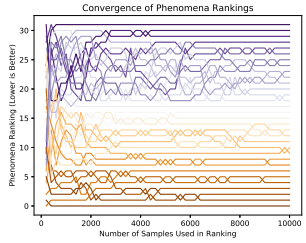
Figure D.89: System 35



(a) Representative

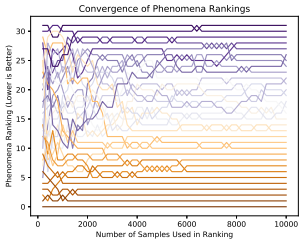


(b) Uniform Fidelity

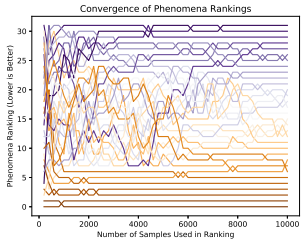


(c) Triangular Fidelity

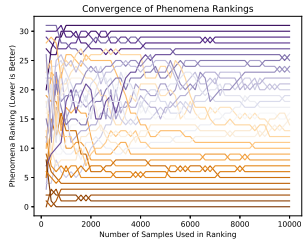
Figure D.90: System 36



(a) Representative

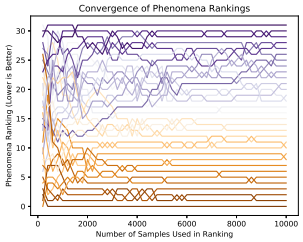


(b) Uniform Fidelity

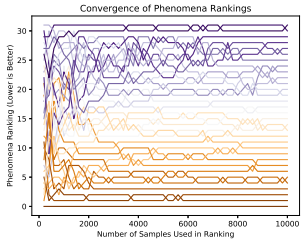


(c) Triangular Fidelity

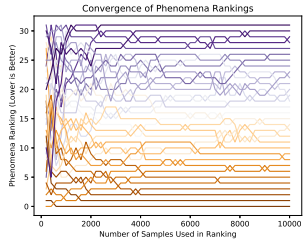
Figure D.91: System 37



(a) Representative

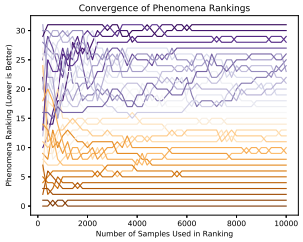


(b) Uniform Fidelity

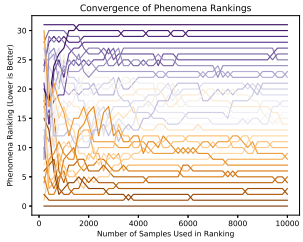


(c) Triangular Fidelity

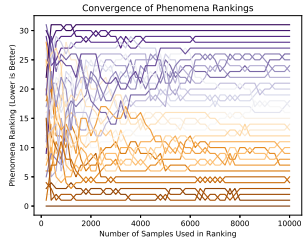
Figure D.92: System 38



(a) Representative

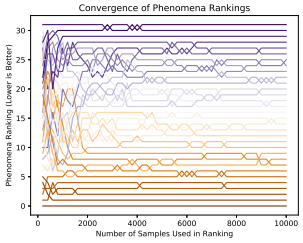


(b) Uniform Fidelity

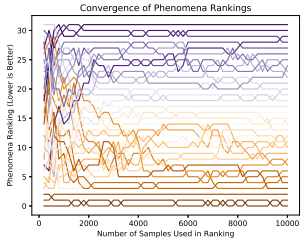


(c) Triangular Fidelity

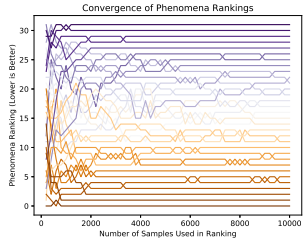
Figure D.93: System 39



(a) Representative

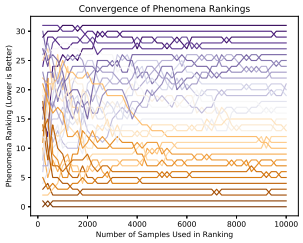


(b) Uniform Fidelity

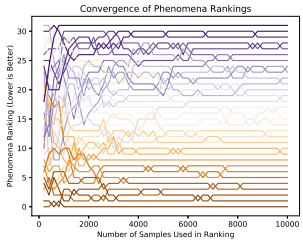


(c) Triangular Fidelity

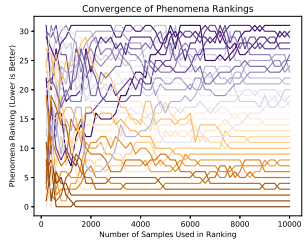
Figure D.94: System 40



(a) Representative

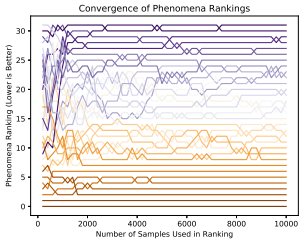


(b) Uniform Fidelity

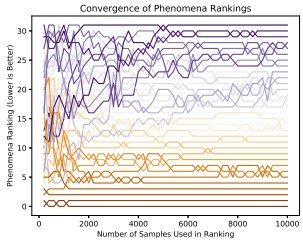


(c) Triangular Fidelity

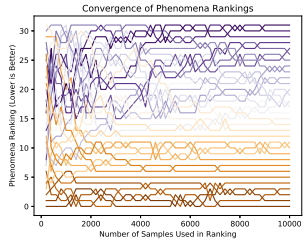
Figure D.95: System 41



(a) Representative

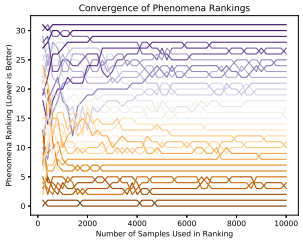


(b) Uniform Fidelity

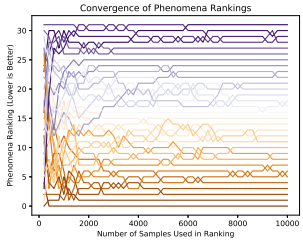


(c) Triangular Fidelity

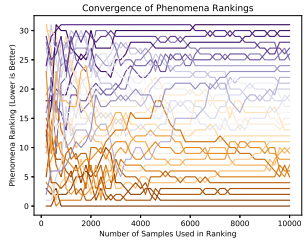
Figure D.96: System 42



(a) Representative

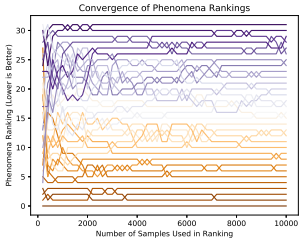


(b) Uniform Fidelity

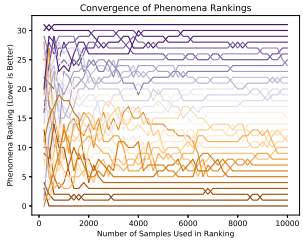


(c) Triangular Fidelity

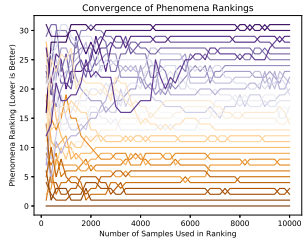
Figure D.97: System 43



(a) Representative

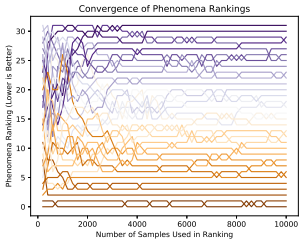


(b) Uniform Fidelity

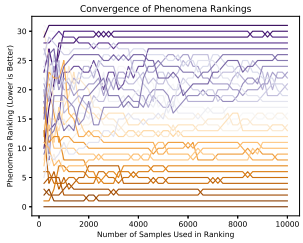


(c) Triangular Fidelity

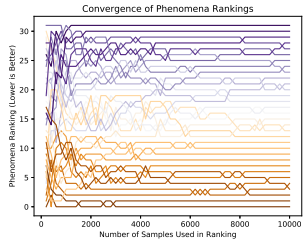
Figure D.98: System 44



(a) Representative

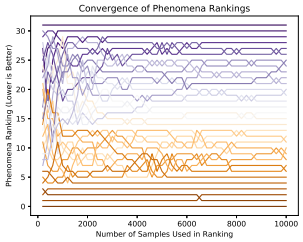


(b) Uniform Fidelity

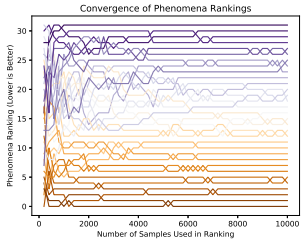


(c) Triangular Fidelity

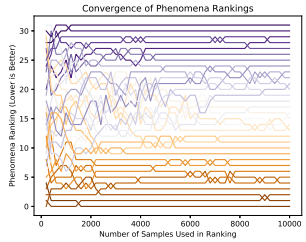
Figure D.99: System 45



(a) Representative

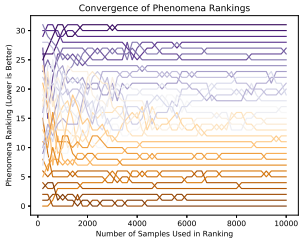


(b) Uniform Fidelity

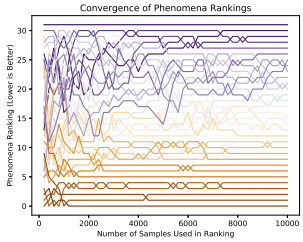


(c) Triangular Fidelity

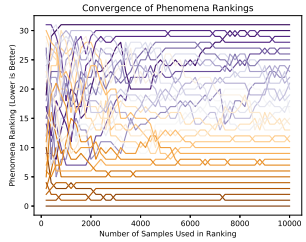
Figure D.100: System 46



(a) Representative

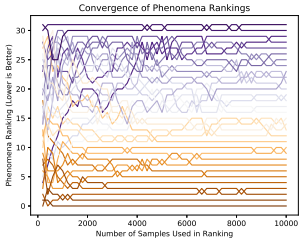


(b) Uniform Fidelity

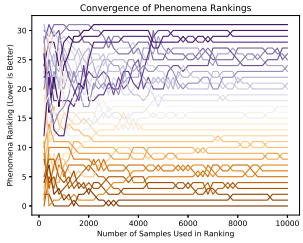


(c) Triangular Fidelity

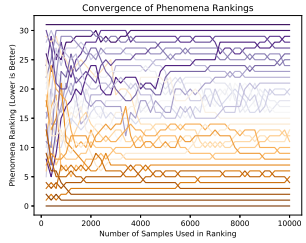
Figure D.101: System 47



(a) Representative

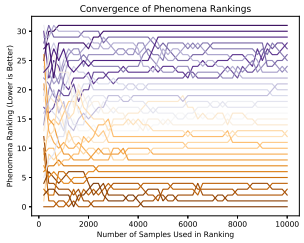


(b) Uniform Fidelity

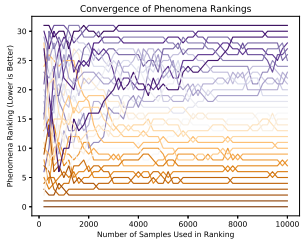


(c) Triangular Fidelity

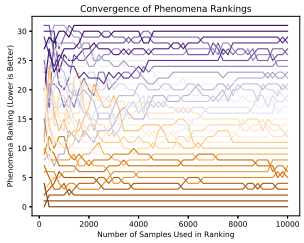
Figure D.102: System 48



(a) Representative

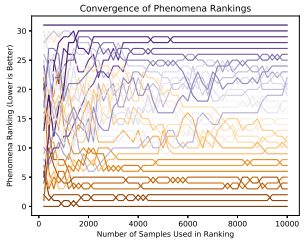


(b) Uniform Fidelity

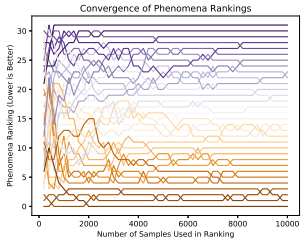


(c) Triangular Fidelity

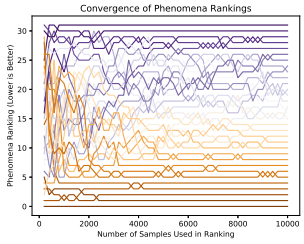
Figure D.103: System 49



(a) Representative



(b) Uniform Fidelity



(c) Triangular Fidelity

D.3 Experiment 4: Effects of Referent Fidelity

D.3.1 Phenomena Ranking Correlation Against Quantitative Measures of Transference

Section 5.2.4 discussed the major results for the imperfect referent experimentation. When discussing the relation between correlation of the phenomena rankings and transference of the referent, it mainly looked at Binary Transference. As this is a necessary condition for quantitative measures of transference, it was expected that there would similarly be little relation between the quantitative measures of transference, Performance and Potential Transference. These relations are shown below in Figure D.104 and Figure D.105. For both of these graphs, there does not seem to be a strong relationship between phenomena correlation and quantitative measures of transference.

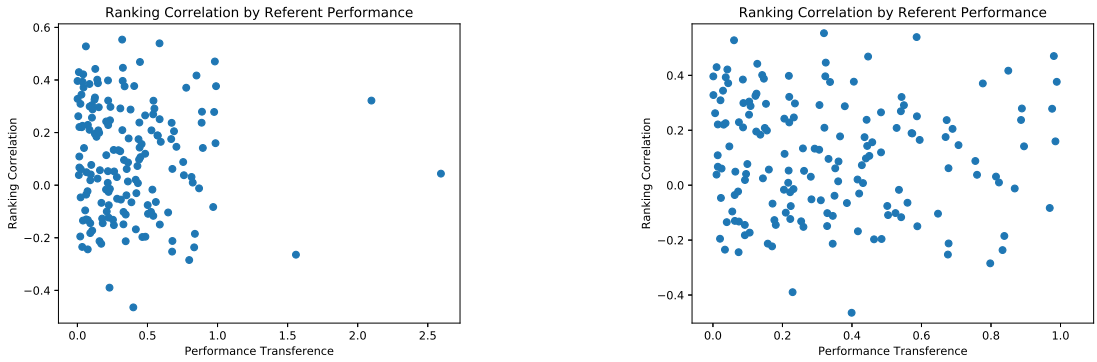


Figure D.104: Graphs showing the relation of Performance Transference of the simplified referent system considered and correlation between phenomena rankings derived on the truth system and rankings derived from the referent system. The right graph is a zoomed version of the left graph to limit the effects of outliers in Performance Transference. No consistent trends are notable.

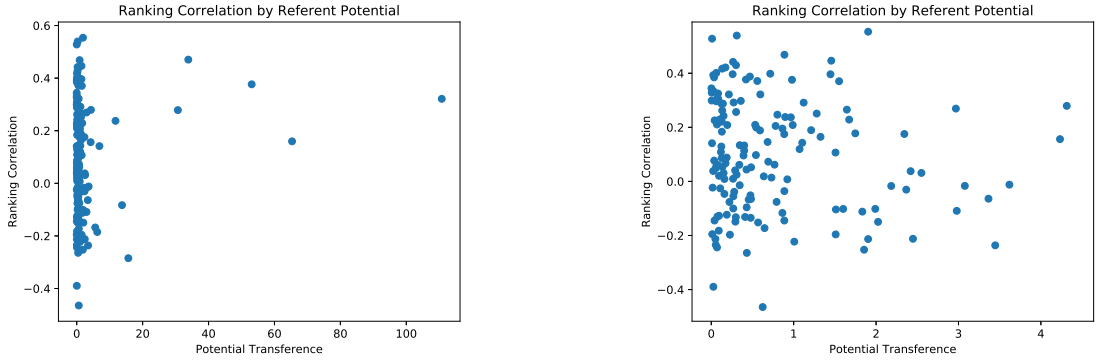
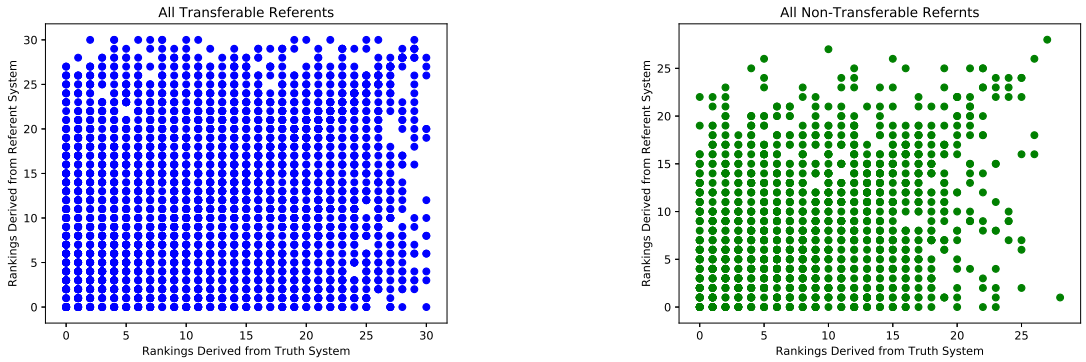


Figure D.105: Graphs showing the relation of Potential Transference of the simplified referent system considered and correlation between phenomena rankings derived on the truth system and rankings derived from the referent system. The right graph is a zoomed version of the left graph to limit the effects of outliers in Potential Transference. No consistent trends are notable.

D.3.2 Individual Results

Section 5.2.4 discussed the major results for the imperfect referent experimentation. These, and the results presented above in Figure D.104 and Figure D.105 looked at correlations between rankings derived with access to the true system and rankings derived only with access to a simplified referent. Results for each of these systems are shown in aggregated form in Figure D.106. When looked at in aggregate, there are no clear distinctions between

the two types of referents.

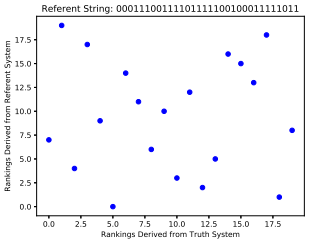


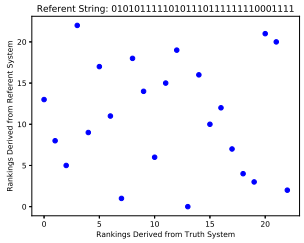
(a) Transferable Referents

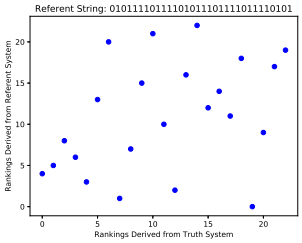
(b) Non-Transferable Referents

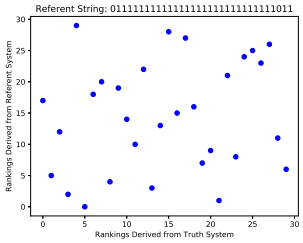
Figure D.106: All ranking comparisons aggregated and separate by if the referent system was capable of training transferable policies.

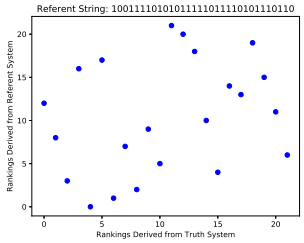
These results are separated out for randomly selected individual referents in the following two sections, classified by whether the referent in question was capable of producing transferable policies or not. When looking at the individual systems, common patterns begin to emerge. First, and least useful, are when the two sets of derived rankings are largely uncorrelated. Second, are systems where the phenomena are separated into clusters of rankings. These show that groups of phenomena have similar levels of criticality, and may be misranked within a grouping. Finally, there are systems where there are a large number of highly correlated phenomena rankings, resulting in a clear line, with the remaining phenomena largely showing no correlation between criticalities derived on the truth system vs. those derived on the false referent. In general, the first set (no correlation) is most common among referents that did not transfer, while the second two sets (clusters and partial correlations) were more common on referent models that did produce transferable policies.

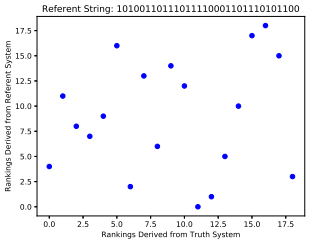

Referent String: 000111001111011111100100011111011
Rankings Derived from Referent System
Rankings Derived from Truth System

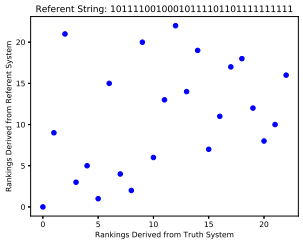

Referent String: 010101111101011101111111100011111
Rankings Derived from Referent System
Rankings Derived from Truth System

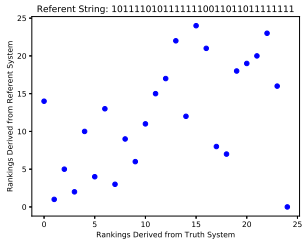

Referent String: 010111101111010111011111011110101
Rankings Derived from Referent System
Rankings Derived from Truth System

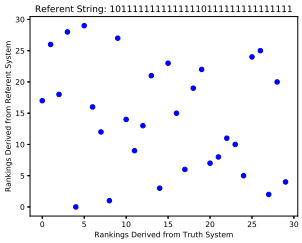

Referent String: 011111111111111111111111111111011
Rankings Derived from Referent System
Rankings Derived from Truth System

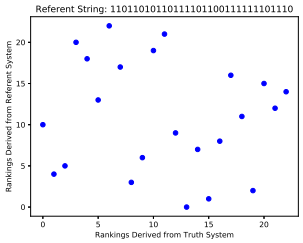

Referent String: 100111101010111101111010111011O110
Rankings Derived from Referent System
Rankings Derived from Truth System

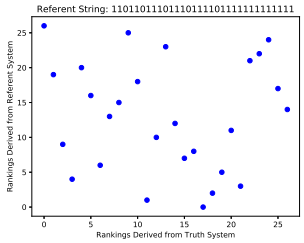

Referent String: 101001101110111100011011010110O
Rankings Derived from Referent System
Rankings Derived from Truth System

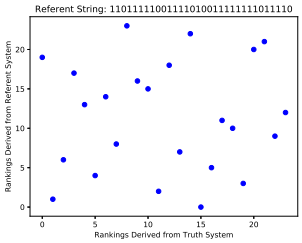

Referent String: 101111001000101111011011111111
Rankings Derived from Referent System
Rankings Derived from Truth System

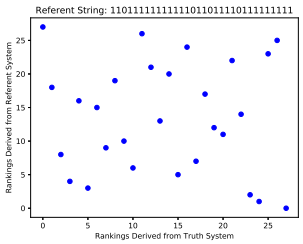

Referent String: 101111010111111100110110111111
Rankings Derived from Referent System
Rankings Derived from Truth System

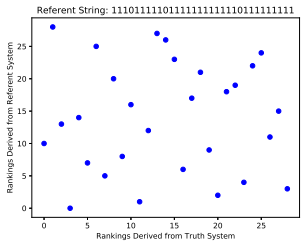

Referent String: 101111111111111011111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

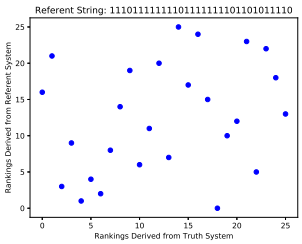

Referent String: 110110101101110110011111101110
Rankings Derived from Referent System
Rankings Derived from Truth System

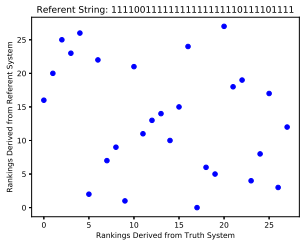

Referent String: 110110111011011101111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

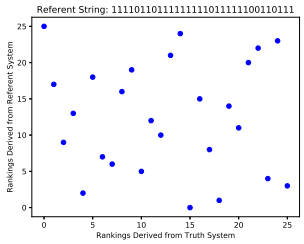

Referent String: 110111110011101001111111011110
Rankings Derived from Referent System
Rankings Derived from Truth System

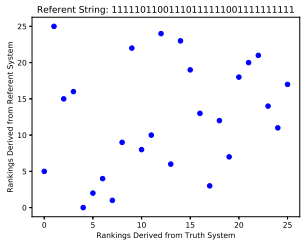

Referent String: 110111111111110110111011111111
Rankings Derived from Referent System
Rankings Derived from Truth System

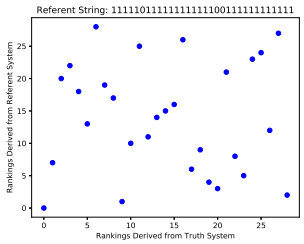

Referent String: 111011111011111111111011111111
Rankings Derived from Referent System
Rankings Derived from Truth System

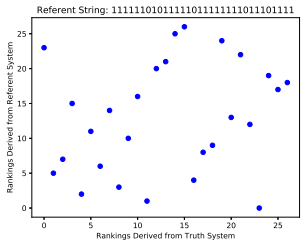

Referent String: 111011111111101111111011011110
Rankings Derived from Referent System
Rankings Derived from Truth System

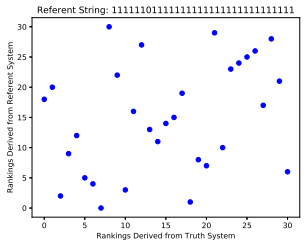

Referent String: 1111001111111111111110111101111
Rankings Derived from Referent System
Rankings Derived from Truth System

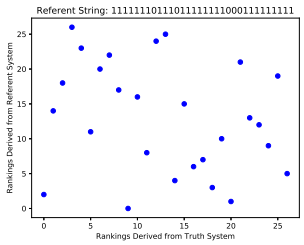

Referent String: 1111101011111111101111100110111
Rankings Derived from Referent System
Rankings Derived from Truth System

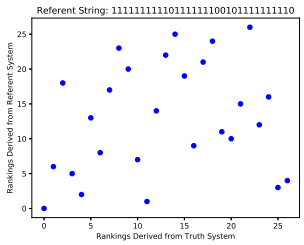

Referent String: 1111101100111011111100111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

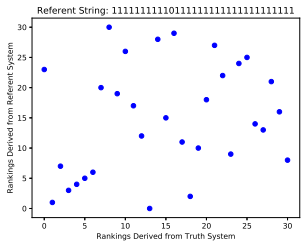

Referent String: 1111101111111111100111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

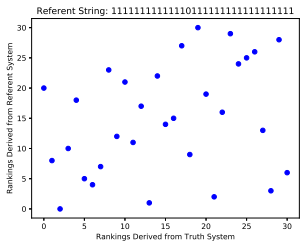

Referent String: 1111110101111101111111011101111
Rankings Derived from Referent System
Rankings Derived from Truth System

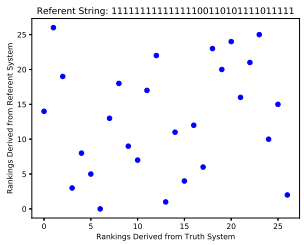

Referent String: 1111110111111111111111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

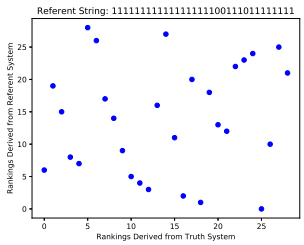

Referent String: 1111111011101111111000111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

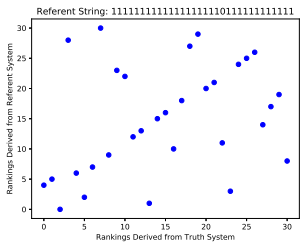

Referent String: 1111111111011111110010111111110
Rankings Derived from Referent System
Rankings Derived from Truth System

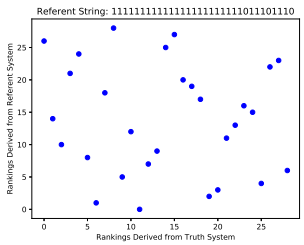

Referent String: 1111111111011111111111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

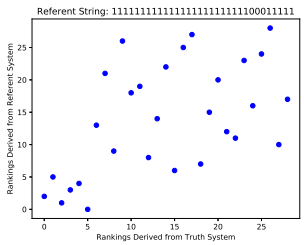

Referent String: 1111111111111011111111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

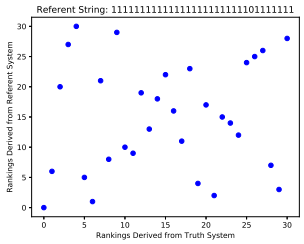

Referent String: 1111111111111100110101111011111
Rankings Derived from Referent System
Rankings Derived from Truth System

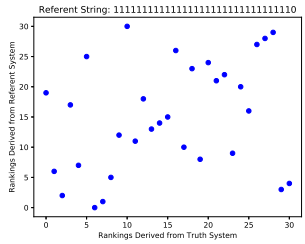

Referent String: 1111111111111111001101111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

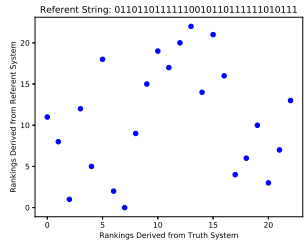

Referent String: 1111111111111111011111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

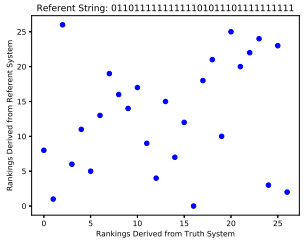

Referent String: 1111111111111111111111011101110
Rankings Derived from Referent System
Rankings Derived from Truth System

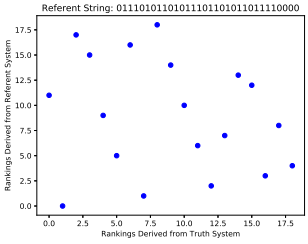

Referent String: 1111111111111111111111100011111
Rankings Derived from Referent System
Rankings Derived from Truth System

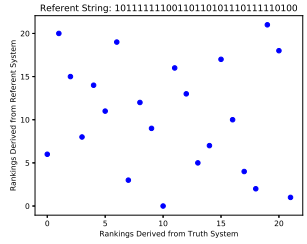

Referent String: 11111111111111111111111101111111
Rankings Derived from Referent System
Rankings Derived from Truth System

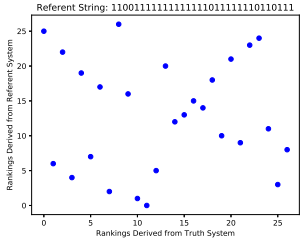

Referent String: 11111111111111111111111111111110
Rankings Derived from Referent System
Rankings Derived from Truth System

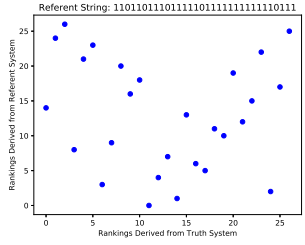

Referent String: 01101101111110010110111111010111
Rankings Derived from Referent System
Rankings Derived from Truth System

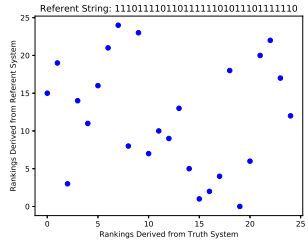

Referent String: 01101111111111101011011111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

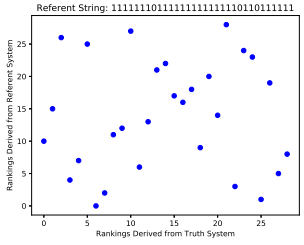

Referent String: 01110101101011101101011110000
Rankings Derived from Referent System
Rankings Derived from Truth System

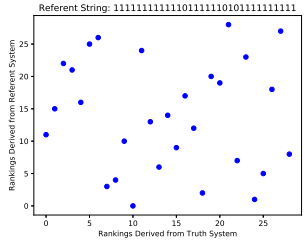

Referent String: 10111111100110110101110111100
Rankings Derived from Referent System
Rankings Derived from Truth System

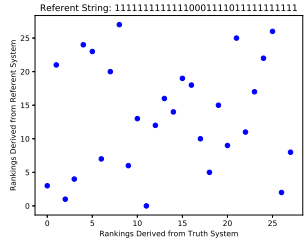

Referent String: 11001111111111111011111110110111
Rankings Derived from Referent System
Rankings Derived from Truth System

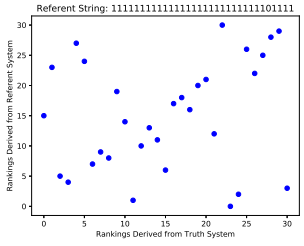

Referent String: 11011011101111101111111111110111
Rankings Derived from Referent System
Rankings Derived from Truth System

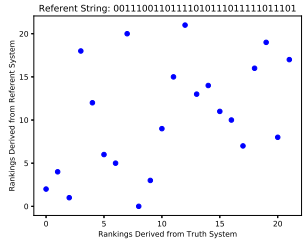

Referent String: 11101111011011111010111011111110
Rankings Derived from Referent System
Rankings Derived from Truth System

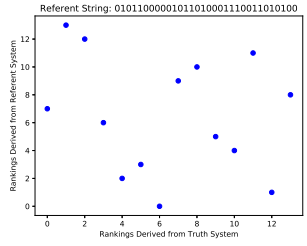

Referent String: 11111110111111111111110110111111
Rankings Derived from Referent System
Rankings Derived from Truth System

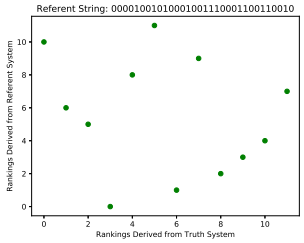

Referent String: 11111111111011111010111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

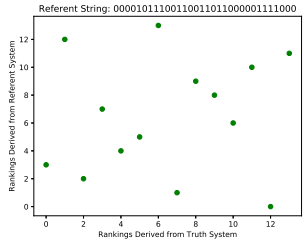

Referent String: 11111111111100011101111111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

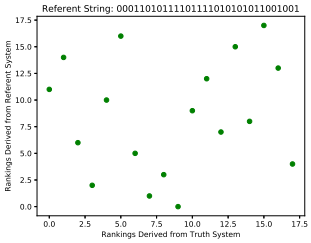

Referent String: 11111111111111111111111101111111
Rankings Derived from Referent System
Rankings Derived from Truth System

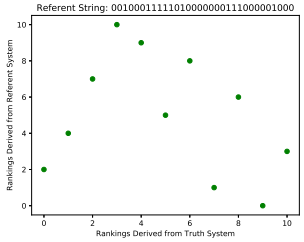

Referent String: 00111001101111010111011111011101
Rankings Derived from Referent System
Rankings Derived from Truth System

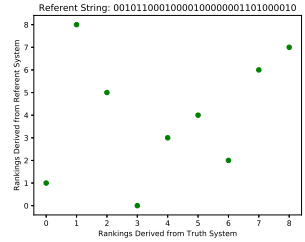

Referent String: 01011000001011010001110011010100
Rankings Derived from Referent System
Rankings Derived from Truth System

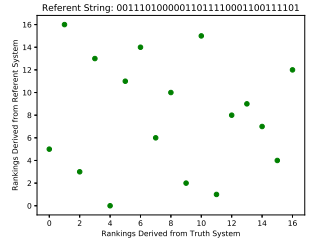

Referent String: 0000100101000100111000110011001 10
Rankings Derived from Referent System
Rankings Derived from Truth System

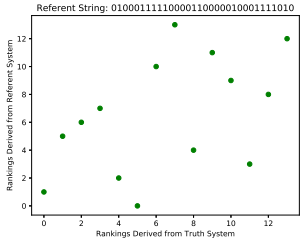

Referent String: 0000101110011001101100000111 1000
Rankings Derived from Referent System
Rankings Derived from Truth System

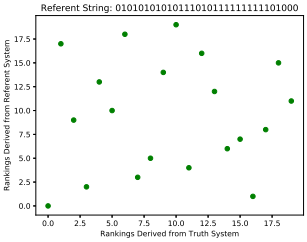

Referent String: 0001101011110111010101011001001
Rankings Derived from Referent System
Rankings Derived from Truth System

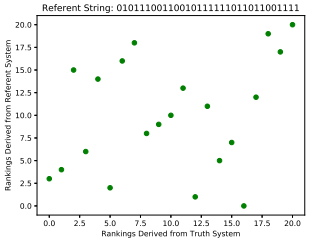

Referent String: 0010001111101000000011100000 1000
Rankings Derived from Referent System
Rankings Derived from Truth System

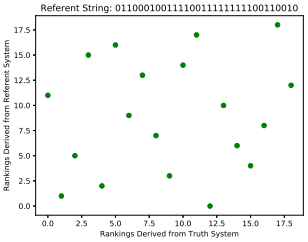

Referent String: 0010110001000010000000110100 0010
Rankings Derived from Referent System
Rankings Derived from Truth System

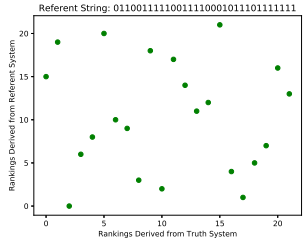

Referent String: 0011101000001101111000110011 1101
Rankings Derived from Referent System
Rankings Derived from Truth System

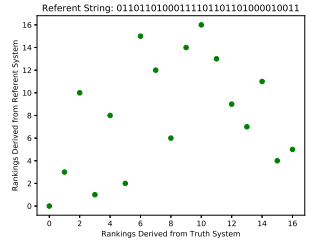

Referent String: 0100011111000011000001000111 1010
Rankings Derived from Referent System
Rankings Derived from Truth System

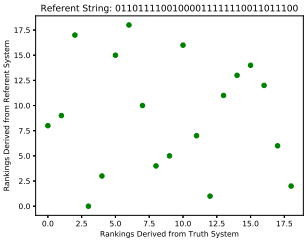

Referent String: 0101010101011101011111111101 1000
Rankings Derived from Referent System
Rankings Derived from Truth System

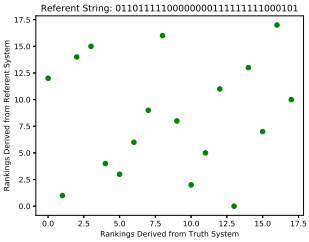

Referent String: 0101110011001011111101101 1111
Rankings Derived from Referent System
Rankings Derived from Truth System

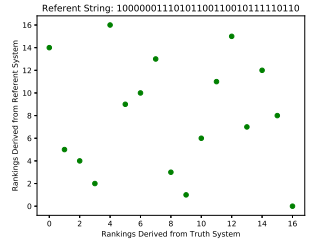

Referent String: 0110001001111001111111110011 0010
Rankings Derived from Referent System
Rankings Derived from Truth System

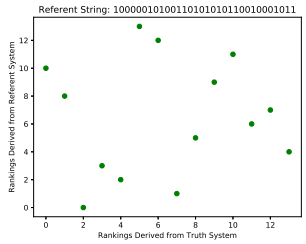

Referent String: 0110011111001111000101110111 1111
Rankings Derived from Referent System
Rankings Derived from Truth System

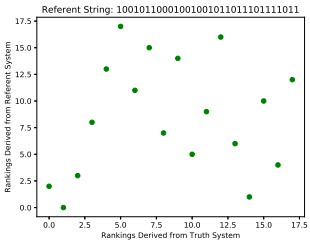

Referent String: 0110110100011110110110100001 0011
Rankings Derived from Referent System
Rankings Derived from Truth System

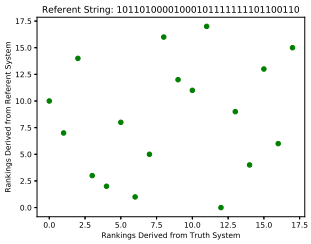

Referent String: 0110111100100001111111001101 1100
Rankings Derived from Referent System
Rankings Derived from Truth System

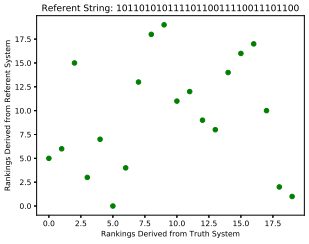

Referent String: 0110111110000000011111111000 0101
Rankings Derived from Referent System
Rankings Derived from Truth System

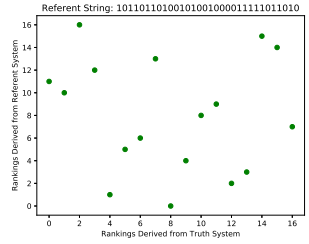

Referent String: 1000000111010110011001011110 1110
Rankings Derived from Referent System
Rankings Derived from Truth System

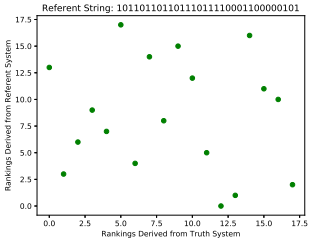

Referent String: 10000010100110101010110010001011
Rankings Derived from Referent System
Rankings Derived from Truth System

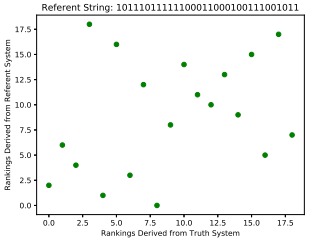

Referent String: 10010110001001001011011101111011
Rankings Derived from Referent System
Rankings Derived from Truth System

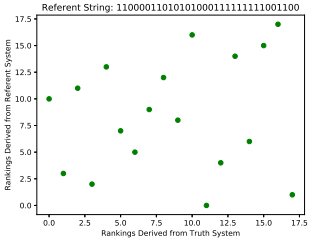

Referent String: 10110100001000101111111101100110
Rankings Derived from Referent System
Rankings Derived from Truth System

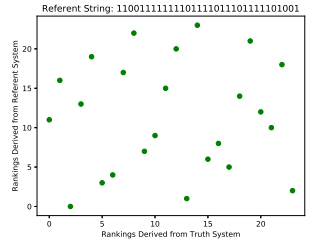

Referent String: 10110101011110110011110011101100
Rankings Derived from Referent System
Rankings Derived from Truth System

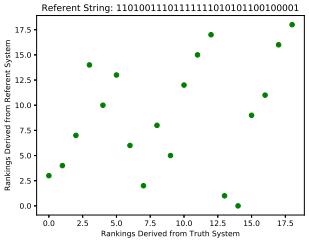

Referent String: 10110110100101001000011111011010
Rankings Derived from Referent System
Rankings Derived from Truth System

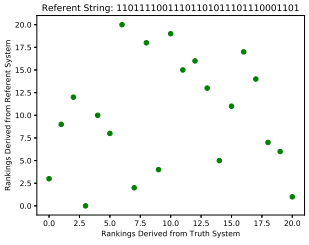

Referent String: 10110110110111011110001100000101
Rankings Derived from Referent System
Rankings Derived from Truth System

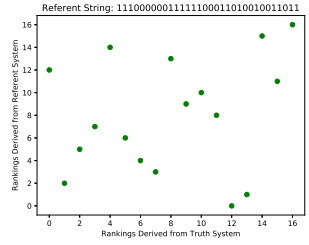

Referent String: 10111011111100011000100111001011
Rankings Derived from Referent System
Rankings Derived from Truth System

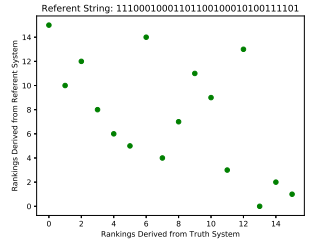

Referent String: 11000011010101000111111111001100
Rankings Derived from Referent System
Rankings Derived from Truth System

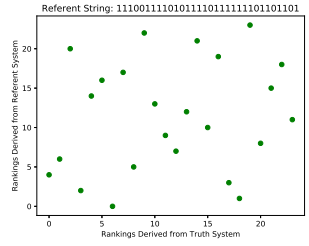

Referent String: 11001111111011101111011111101001
Rankings Derived from Referent System
Rankings Derived from Truth System

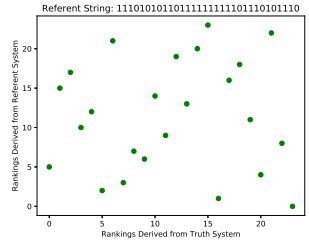

Referent String: 11010011101111111010101100100001
Rankings Derived from Referent System
Rankings Derived from Truth System

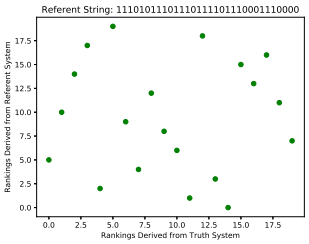

Referent String: 11011110011101101011101110001101
Rankings Derived from Referent System
Rankings Derived from Truth System

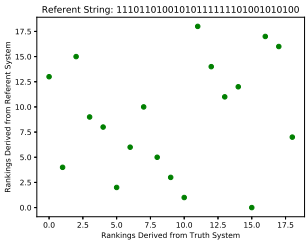

Referent String: 11100000011111100011010010011011
Rankings Derived from Referent System
Rankings Derived from Truth System

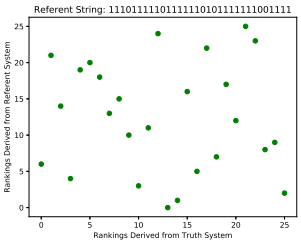

Referent String: 11100010001101100100010100111101
Rankings Derived from Referent System
Rankings Derived from Truth System

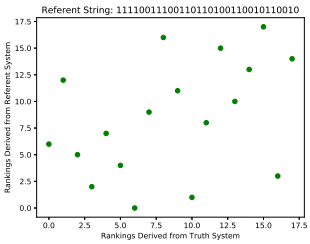

Referent String: 11100111101011110111111101101101
Rankings Derived from Referent System
Rankings Derived from Truth System

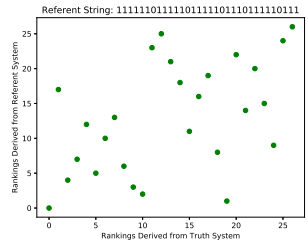

Referent String: 11101010110111111111101101011110
Rankings Derived from Referent System
Rankings Derived from Truth System

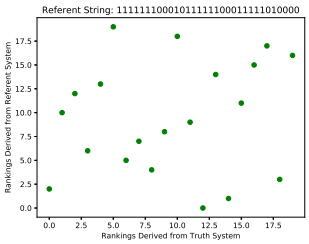

Referent String: 11101011101110111101110001110000
Rankings Derived from Referent System
Rankings Derived from Truth System

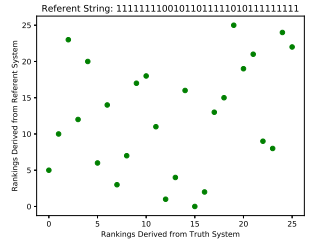

Referent String: 11101101001010111111010010101000
Rankings Derived from Referent System
Rankings Derived from Truth System

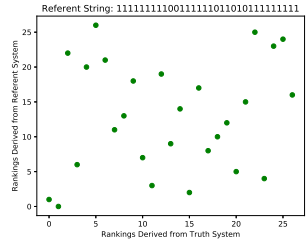

Referent String: 11101111101111101011111001111
Rankings Derived from Referent System
Rankings Derived from Truth System

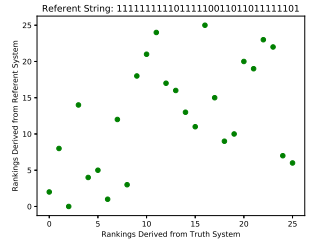

Referent String: 11110011100110110100110010110010
Rankings Derived from Referent System
Rankings Derived from Truth System

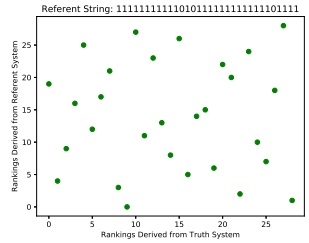

Referent String: 11111101111101111101111110111
Rankings Derived from Referent System
Rankings Derived from Truth System

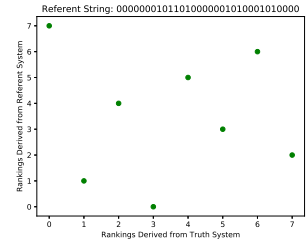

Referent String: 11111110001011111000111110100000
Rankings Derived from Referent System
Rankings Derived from Truth System

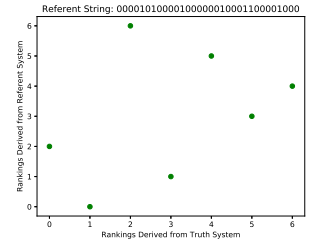

Referent String: 11111111001011011110101111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

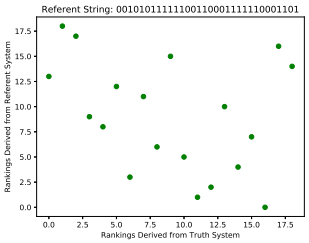

Referent String: 11111111100111111011010111111111
Rankings Derived from Referent System
Rankings Derived from Truth System

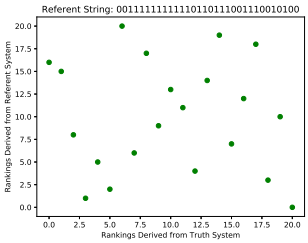

Referent String: 11111111110111110011011011111101
Rankings Derived from Referent System
Rankings Derived from Truth System

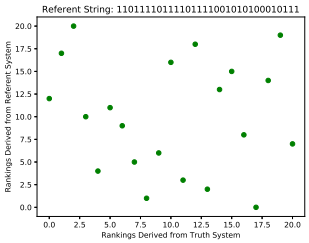

Referent String: 11111111111010111111111111101111
Rankings Derived from Referent System
Rankings Derived from Truth System


Referent String: 00000001011010000001010001010000
Rankings Derived from Referent System
Rankings Derived from Truth System


Referent String: 00001010000100000010001100001000
Rankings Derived from Referent System
Rankings Derived from Truth System


Referent String: 00101011111100110001111110001101
Rankings Derived from Referent System
Rankings Derived from Truth System


Referent String: 00111111111101101110011100100100
Rankings Derived from Referent System
Rankings Derived from Truth System


Referent String: 11011110111101110010101000010111
Rankings Derived from Referent System
Rankings Derived from Truth System

REFERENCES

[1] F. A. Administration, *PART 60—FLIGHT SIMULATION TRAINING DEVICE INITIAL AND CONTINUING QUALIFICATION AND USE*. Electronic Code of Federal Regulations, 2006.

[2] C. E. Agüero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo, E. Krotkov, and G. Pratt, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 494–506, 2015.

[3] "Airplane simulator qualification," United States Federal Aviation Administration, Tech. Rep. AC 120-40B, 1991.

[4] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Proceedings of International Conference on Robotics and Automation*, vol. 4, 1997, 3557–3564 vol.4.

[5] A. Atkinson, "Dt-optimum designs for model discrimination and parameter estimation," *Journal of Statistical Planning and Inference*, vol. 138, no. 1, pp. 56 –64, 2008, International Conference on Design of Experiments (ICODOE).

[6] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 908–918.

[7] S. Bouarfa, H. A. Blom, R. Curran, and M. H. Everdij, "Agent-based modeling and simulation of emergent behavior in air transportation," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, p. 15, 2013.

[8] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4243–4250.

[9] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] G. E. P. Box, "Science and statistics," *Journal of the American Statistical Association*, vol. 71, no. 356, pp. 791–799, 1976.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, 2016. eprint: arXiv:1606.01540.

[12] E. Burnett, in, ser. Guidance, Navigation, and Control and Co-located Conferences. American Institute of Aeronautics and Astronautics, 2008, ch. A Proposed Model Fidelity Scale. 0.

[13] E. Calvano, G. Calzolari, V. Denicolò, and S. Pastorello, "Algorithmic pricing what implications for competition policy?" *Review of Industrial Organization*, 2019.

[14] W. K. V. Chan, Y. Son, and C. M. Macal, "Agent-based simulation tutorial - simulation of emergent behavior and differences between agent-based simulation and discrete-event simulation," in *Proceedings of the 2010 Winter Simulation Conference*, 2010, pp. 135–150.

[15] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8973–8979.

[16] K. A. Clarke, "Nonparametric model discrimination in international relations," *Journal of Conflict Resolution*, vol. 47, no. 1, pp. 72–93, 2003. eprint: https://doi.org/10.1177/0022002702239512.

[17] K. A. Clarke and C. S. Signorino, "Discriminating methods: Tests for non-nested discrete choice models," *Political Studies*, vol. 58, no. 2, pp. 368–388, 2010. eprint: https://doi.org/10.1111/j.1467-9248.2009.00813.x.

[18] S. Cook, A. Dietrich, L. Hook, and A. Lacher, "Promoting autonomy design and operations in aviation," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–9.

[19] A. W. Cox, *Fidelity assessment for model selection (FAMS): A framework for initial comparison of multifidelity modeling options*. Georgia Institute of Technology, 2019.

[20] M. Cutler and J. P. How, "Autonomous drifting using simulation-aided reinforcement learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5442–5448.

[21] M. Cutler, T. J. Walsh, and J. P. How, "Real-world reinforcement learning via multifidelity simulators," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 655–671, 2015.

[22] A. Drogoul, D. Vanbergue, and T. Meurisse, "Multi-agent based simulation: Where are the agents?" In *Proceedings of the 3rd International Conference on Multi-agent-based Simulation II*, ser. MABS'02, Bologna, Italy: Springer-Verlag, 2003, pp. 1–15, ISBN: 3-540-00607-9.

[23] J. A. Estefan *et al.*, "Survey of model-based systems engineering (mbse) methodologies," *Incose MBSE Focus Group*, vol. 25, no. 8, pp. 1–12, 2007.

[24] K. M. Fahey and M. J. Miller, "Unmanned systems integrated roadmap fy 2017-2042," United States Department of Defense Office of the Secretary of Defense, Tech. Rep., 2018.

[25] *Fidelity, n.* In *Oxford English Dictionary Online*, Oxford University Press.

[26] R. L. Figueroa, Q. Zeng-Treitler, S. Kandula, and L. H. Ngo, "Predicting sample size required for classification performance," *BMC Medical Informatics and Decision Making*, vol. 12, no. 1, p. 8, 2012.

[27] F. Fleurey, B. Baudry, R. France, and S. Ghosh, "A generic approach for automatic model composition," in *Models in Software Engineering*, H. Giese, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 7–15, ISBN: 978-3-540-69073-3.

[28] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: From architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[29] G. Fontaine and O. Hammami, "Automatic model search for system model composition," in *2018 IEEE International Systems Engineering Symposium (ISSE)*, 2018, pp. 1–7.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680.

[31] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, 2014. arXiv: `1412.6572 [stat.ML]`.

[32] V. Grimm, E. Revilla, U. Berger, F. Jeltsch, W. M. Mooij, S. F. Railsback, H.-H. Thulke, J. Weiner, T. Wiegand, and D. L. DeAngelis, "Pattern-oriented model-

ing of agent-based complex systems: Lessons from ecology," *Science*, vol. 310, no. 5750, pp. 987–991, 2005. eprint: http://science.sciencemag.org/content/310/5750/987.full.pdf.

[33] E. D. Grober, S. J. Hamstra, K. R. Wanzel, R. K. Reznick, E. D. Matsumoto, R. S. Sidhu, and K. A. Jarvi, "The educational impact of bench model fidelity on the acquisition of technical skill: The use of clinically relevant outcome measures," *Annals of surgery*, vol. 240, no. 2, p. 374, 2004.

[34] D. C. Gross *et al.*, "Report from the fidelity implementation study group," in *Fall Simulation Interoperability Workshop Papers*, 1999.

[35] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Advanced Robotics*, vol. 21, no. 13, pp. 1521–1544, 2007. eprint: https://www.tandfonline.com/doi/pdf/10.1163/156855307782148550.

[36] A. Günay and P. Yolum, "Structural and semantic similarity metrics for web service matchmaking," in *E-Commerce and Web Technologies*, G. Psaila and R. Wagner, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 129–138, ISBN: 978-3-540-74563-1.

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 1861–1870.

[38] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

[39] O. Hammami, "Multiobjective optimization of collaborative process for modeling and simulation - ¡q, r, t¿," in *2015 IEEE International Symposium on Systems Engineering (ISSE)*, 2015, pp. 446–453.

[40] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16, Phoenix, Arizona: AAAI Press, 2016, 2094–2100.

[41] K. A. Hawick, C. J. Scogings, and H. A. James, "Defensive spiral emergence in a predator-prey model," *Complexity International*, vol. 12, no. msid37, R. Stonier, Q. Han, and W. Li, Eds., pp. 1–10, 2008, ISSN 1320-0682.

[42] R. T. Hays, "Simulator fidelity: A concept paper," ARMY RESEARCH INST FOR THE BEHAVIORAL and SOCIAL SCIENCES ALEXANDRIA VA, Tech. Rep., 1980.

[43] R. T. Hays and M. J. Singer, *Simulation Fidelity in Training System Design*. Springer, New York, NY, 1989.

[44] T. A. N. Heirung, T. L. Santos, and A. Mesbah, "Model predictive control with active learning for stochastic systems with structural model uncertainty: Online model discrimination," *Computers and Chemical Engineering*, vol. 128, pp. 128 –140, 2019.

[45] J. Hirtz, R. B. Stone, D. A. McAdams, S. Szykman, and K. L. Wood, "A functional basis for engineering design: Reconciling and evolving previous efforts," *Research in Engineering Design*, vol. 13, no. 2, pp. 65–82, 2002.

[46] J. H. Holland, *Emergence: From chaos to order*. OUP Oxford, 2000.

[47] O. T. Holland, "Partitioning method for emergent behavior systems modeled by agent-based simulations," Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2016-03-11, Ph.D. dissertation, 2012, p. 267, ISBN: 9781267890504.

[48] R. Horn, "Statistical methods for model discrimination. applications to gating kinetics and permeation of the acetylcholine receptor channel," *Biophysical Journal*, vol. 51, no. 2, pp. 255 –263, 1987.

[49] S. C. S. Hunter, D. D. C. Jensen, I. Y. I. Tumer, and C. Hoyle, "The Impact of Abstraction and Fidelity Levels on the Usefulness of Early System Functional Models," *Proceedings of the ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 1B-2016, pp. 1–12, 2016.

[50] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2019.

[51] R. L. Iman, J. C. Helton, and J. E. Campbell, "An approach to sensitivity analysis of computer models: Part i—introduction, input variable selection and preliminary variable assessment," *Journal of Quality Technology*, vol. 13, no. 3, pp. 174–183, 1981. eprint: https://doi.org/10.1080/00224065.1981.11978748.

[52] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green,

L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel, "Human-level performance in 3d multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, 2019. eprint: https://science.sciencemag.org/content/364/6443/859.full.pdf.

[53] S. James and E. Johns, *3d simulation for robot arm control with deep q-learning*, 2016. arXiv: 1609.03759 [cs.RO].

[54] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[55] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 12 519–12 530.

[56] E. Johnson, D. Schrage, and G. Vachtsevanos, "Software enabled control experiments with university operated unmanned aircraft," in, ser. Infotech@Aerospace Conferences. American Institute of Aeronautics and Astronautics, 2005, 0.

[57] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[58] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[59] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Mach. Learn.*, vol. 49, no. 2–3, 209–232, Nov. 2002.

[60] D. L. Key *et al.*, "Fidelity of simulation for pilot training," *AGARD Advisory Report*, vol. 159, 1980.

[61] H. J. Kim, M. I. Jordan, S. Sastry, and A. Y. Ng, "Autonomous helicopter flight via reinforcement learning," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds., MIT Press, 2004, pp. 799–806.

[62] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. arXiv: 1412.6980 [cs.LG].

[63] K. Kiriakidis and D. F. Gordon-Spears, "Formal modeling and supervisory control of reconfigurable robot teams," in *Formal Approaches to Agent-Based Systems*, M. G. Hinchey, J. L. Rash, W. F. Truszkowski, C. Rouff, and D. Gordon-Spears, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 92–102, ISBN: 978-3-540-45133-4.

[64] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013. eprint: https://doi.org/10.1177/0278364913495721.

[65] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, 2149–2154 vol.3.

[66] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. Müller, Eds., MIT Press, 2000, pp. 1008–1014.

[67] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[68] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 204–211.

[69] N. E. Lane and E. A. Alluisi, "Fidelity and validity in distributed interactive simulation: Questions and answers," INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA, Tech. Rep., 1992.

[70] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Communications*, vol. 10, no. 1, 2019.

[71] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, *Continuous control with deep reinforcement learning*, 2015. arXiv: 1509.02971 [cs.LG].

[72] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, T. K. Marks, and R. Chellappa, "Fast object localization and pose estimation in heavy clutter for robotic bin picking," *The International Journal of Robotics Research*, vol. 31, no. 8, pp. 951–973, 2012. eprint: https://doi.org/10.1177/0278364911436018.

[73] A. J. McLane, C. Semeniuk, G. J. McDermid, and D. J. Marceau, "The role of agent-based models in wildlife ecology and management," *Ecological Modelling*, vol. 222, no. 8, pp. 1544 –1556, 2011.

[74] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, *Active domain randomization*, 2019. arXiv: 1904.04762 [cs.LG].

[75] M. D. Mesarovic and Y. Takahara, *General systems theory: mathematical foundations*. Academic press, 1975, vol. 113.

[76] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, New York, NY, USA: JMLR.org, 2016, 1928–1937.

[77] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, 2013. arXiv: 1312.5602 [cs.LG].

[78] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–33, Feb. 2015.

[79] J. C. Mogul, "Emergent (mis) behavior vs. complex software systems," in *ACM SIGOPS Operating Systems Review*, ACM, vol. 40, 2006, pp. 293–304.

[80] J. I. Myung and M. A. Pitt, "Optimal experimental design for model discrimination," *Psychological review*, vol. 116, no. 3, pp. 499–518, 2009, 19618983[pmid].

[81] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ser. ICML '99, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, 278–287, ISBN: 1558606122.

[82] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, Matthias, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *arXiv preprint*, 2019.

[83] D. K. Pace, "Issues related to quantifying simulation validation," in *Proceedings of SCSC*, vol. 1, 2001, pp. 15–19.

[84] J. H. Panchal, C. J. Paredis, J. K. Allen, and F. Mistree, "A value-of-information based approach to simulation model refinement," *Engineering Optimization*, vol. 40, no. 3, pp. 223–251, 2008. eprint: https://doi.org/10.1080/03052150701690764.

[85] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," *SIAM Review*, vol. 60, no. 3, pp. 550–591, 2018. eprint: https://doi.org/10.1137/16M1082469.

[86] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7, pp. 1180 –1190, 2008, Progress in Modeling, Theory, and Application of Computational Intelligenc.

[87] P. Petridis, I. Dunwell, S. De Freitas, and D. Panzoli, "An engine selection methodology for high fidelity serious games," in *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on*, IEEE, 2010, pp. 27–34.

[88] B. Planche, Z. Wu, K. Ma, S. Sun, S. Kluckner, T. Chen, A. Hutter, S. I. Zakharov, H. Kosch, and J. Ernst, "Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5d recognition," *2017 International Conference on 3D Vision (3DV)*, pp. 1–10, 2017.

[89] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," in *International Conference on Learning Representations*, 2018.

[90] A. S. Polydoros, L. Nalpantidis, and V. Krüger, "Real-time deep learning of robotic manipulator inverse dynamics," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3442–3448.

[91] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[92] L. Pronzato and W. G. Müller, "Design of computer experiments: Space filling and beyond," *Statistics and Computing*, vol. 22, no. 3, pp. 681–701, 2012.

[93] R. Radhakrishnan and D. A. McAdams, "A Methodology for Model Selection in Engineering Design," *Journal of Mechanical Design*, vol. 127, no. 3, pp. 378–387, May 2005. eprint: https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/127/3/378/5601361/378_1.pdf.

[94] L. B. Rainey and A. Tolk, *Modeling and simulation support for system of systems engineering applications*. John Wiley & Sons, 2015.

[95] S. Rank, C. Hammel, T. Schmidt, J. Müller, A. Wenzel, R. Lasch, and G. Schneider, "The correct level of model complexity in semiconductor fab simulation — lessons learned from practice," in *2016 27th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, 2016, pp. 133–139.

[96] S Robinson, "Conceptual modelling for simulation part i: Definition and requirements," *Journal of the Operational Research Society*, vol. 59, no. 3, pp. 278–290, 2008. eprint: https://doi.org/10.1057/palgrave.jors.2602368.

[97] S. Robinson, "Conceptual modelling for simulation part i: Definition and requirements," *Journal of the Operational Research Society*, vol. 59, pp. 278–290, Mar. 2008.

[98] M. Roza, J. Voogd, and D. Sebalj, "The Generic Methodology for Verification and Validation to support acceptance of models , simulations and data," 2012.

[99] Z. Roza, D. Gross, and S. Harmon, "Report out of the fidelity experimentation isg," *complexity*, vol. 4, no. 11, p. 12, 2000.

[100] Z. C. Roza, *Simulation fidelity theory and practice: a unified approach to defining, specifying and measuring the realism of simulations*. DUP Science, 2004.

[101] S. Seifert, P. Meyer, C. Ramee, E. Evans, W. Roberts, K. Griendling, and D. Mavris, "Arcs: A unified environment for autonomous robot control and simulation," in *OCEANS 2017 - Anchorage*, 2017, pp. 1–8.

[102] O. G. Selfridge, R. S. Sutton, and A. G. Barto, "Training and tracking in robotics," in *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, ser. IJCAI'85, Los Angeles, California: Morgan Kaufmann Publishers Inc., 1985, 670–672, ISBN: 0934613028.

[103] S. Shah, D. Dey, C. Lovett, and A. Kapoor, *Airsim: High-fidelity visual and physical simulation for autonomous vehicles*, 2017. arXiv: 1705.05065 [cs.RO].

[104] N. T. Siebel and G. Sommer, "Evolutionary reinforcement learning of artificial neural networks," *International Journal of Hybrid Intelligent Systems*, vol. 4, pp. 171–183, 2007, 3.

[105] P. O. Siebers, C. M. Macal, J Garnett, D Buxton, and M Pidd, "Discrete-event simulation is dead, long live agent-based simulation!" *Journal of Simulation*, vol. 4, no. 3, pp. 204–210, 2010. eprint: https://doi.org/10.1057/jos.2010.14.

[106] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International*

Conference on Machine Learning, E. P. Xing and T. Jebara, Eds., ser. Proceedings of Machine Learning Research, vol. 32, Bejing, China: PMLR, 2014, pp. 387–395.

[107] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[108] M. W. Spong, "The swing up control problem for the acrobot," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 49–55, 1995.

[109] "Standard Practice for Methods to Safely Bound Flight Behavior of Unmanned Aircraft Systems Containing Complex Functions," ASTM International, West Conshohocken, PA, Standard, 2017.

[110] G. Stevens and S. Atamturktur, "Mitigating error and uncertainty in partitioned analysis: A review of verification, calibration and validation methods for coupled simulations," *Archives of Computational Methods in Engineering*, vol. 24, no. 3, pp. 557–571, 2017.

[111] Y. Sun, D. Wierstra, T. Schaul, and J. Schmidhuber, "Efficient natural evolution strategies," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '09, Montreal, Québec, Canada: Association for Computing Machinery, 2009, 539–546, ISBN: 9781605583259.

[112] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018, ISBN: 0262039249.

[113] C. Szabo and Y. M. Teo, "An integrated approach for the validation of emergence in component-based simulation models," in *Proceedings of the winter simulation conference*, Winter Simulation Conference, 2012, p. 242.

[114] ——, "Formalization of weak emergence in multiagent systems," *ACM Trans. Model. Comput. Simul.*, vol. 26, no. 1, 6:1–6:25, Sep. 2015.

[115] C. Szabo, Y. M. Teo, and G. K. Chengleput, "Understanding complex systems: Using interaction as a measure of emergence," *Proceedings - Winter Simulation Conference*, vol. 2015-Janua, pp. 207–218, 2015. arXiv: arXiv:1011.1669v3.

[116] "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," Society of Automotive Engineers, Warrendale, PA, Standard, Jun. 2018.

[117] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, no. 56, pp. 1633–1685, 2009.

[118] Y. M. Teo, B. L. Luong, and C. Szabo, "Formalization of emergence in multi-agent systems," in *Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, ACM, 2013, pp. 231–240.

[119] S. B. Thrun, "Efficient exploration in reinforcement learning," Carnegie Melon University, Tech. Rep., 1992.

[120] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[121] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[122] A. Tolk, M. T. K. Koehler, and M. D. Norman, "Epistemological constraints when evaluating ontological emergence with computational complex adaptive systems," in *Unifying Themes in Complex Systems IX: Proceedings of the Ninth International Conference on Complex Systems*, Springer, 2018, pp. 1–10.

[123] A. J. Turner and D. N. Mavris, "Conceptual modeling and validation of a ha/dr scenario using a weighted system decomposition model," in *Winter Simulation Conference (WSC), 2015*, IEEE, 2015, pp. 2487–2498.

[124] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[125] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Phys. Rev.*, vol. 36, pp. 823–841, 5 1930.

[126] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura, and T. Tomiyama, "Asupporting conceptual design based on the function-behavior-state modeler," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 10, no. 4, 275–288, 1996.

[127] D.-J. van der Zee, "Model simplification in manufacturing simulation – review and framework," *Computers & Industrial Engineering*, vol. 127, pp. 1056 –1067, 2019.

[128] A. Varas, M. D. Cornejo, B. A. Toledo, V. Muñoz, J. Rogan, R. Zarama, and J. A. Valdivia, "Resonance, criticality, and emergence in city traffic investigated in cellular automaton models," *Phys. Rev. E*, vol. 80, p. 056 108, 5 2009.

[129] A. Vemula, W. Sun, and J. Bagnell, "Contrasting exploration in parameter and action space: A zeroth-order optimization perspective," in *Proceedings of Machine Learning Research*, K. Chaudhuri and M. Sugiyama, Eds., ser. Proceedings of Machine Learning Research, vol. 89, PMLR, 2019, pp. 2926–2935.

[130] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135 –153, 2018.

[131] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[132] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[133] A. F. Winfield, J. Sa, M.-C. Fernández-Gago, C. Dixon, and M. Fisher, "On formal specification of emergent behaviours in swarm robotic systems," *International Journal of Advanced Robotic Systems*, vol. 2, no. 4, p. 39, 2005. eprint: `https://doi.org/10.5772/5769`.

[134] J. C. F. de Winter, D. Dodou, and M. Mulder, "Training effectiveness of whole body flight simulator motion: A comprehensive meta-analysis," *The International Journal of Aviation Psychology*, vol. 22, no. 2, pp. 164–183, 2012. eprint: `https://doi.org/10.1080/10508414.2012.663247`.

[135] W. Yip and T. Marlin, "The effect of model fidelity on real-time optimization performance," *Computers & Chemical Engineering*, vol. 28, no. 1, pp. 267 –280, 2004, Escape 12.

[136] A. M. Zaremski and J. M. Wing, "Specification matching of software components," *ACM Trans. Softw. Eng. Methodol.*, vol. 6, no. 4, pp. 333–369, Oct. 1997.

[137] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of modeling and simulation*. Academic press, 2000.

[138] F. Zhang, J. Leitner, Z. Ge, M. Milford, and P. Corke, "Adversarial discriminative sim-to-real transfer of visuo-motor policies," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1229–1245, 2019. eprint: `https://doi.org/10.1177/0278364919870227`.

[139] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," in *Australasian Conference on Robotics and Auomation 2015 (ACRA)*, 2015.